Murmeration

For tenor saxophone & live electronics

Kier Hall

Murmeration

for tenor saxophone and live electronics
Duration: c. 17 minutes

Equipment Required

Microphone

MIDI footswitch (or TRS footswitch with adapter; see below)

Computer with Pure Data (vanilla) installed.

Murmeration electronic resource folder. This includes the patch and supporting files.

Pure Data will require the following extensions: list-abs, maxlib, cyclone and freeverb.

Note: this patch uses 5.1 surround sound and therefore the computer's sound card will require 5.1 output. The patch will automatically route the output spatially as required. PD settings may need to be checked to include support for DAC channels 1 to 6.

MIDI Information

The patch requires one MIDI note to be pressed multiple times throughout the performance via a footswitch interface. This can be in the form of a MIDI controller footswitch or a more commonly found 'sustain pedal' (non-latching) footswitch with TRS connector which can be connected using a footswitch-to-MIDI adapter.

Note: the patch is configured to detect any MIDI channel and any note in order to trigger events so there is no further configuration required when setting up the controller.

Preparation

Load *murmeration.pd*.

Go to Preferences > MIDI and select the MIDI controller in Input Devices.

Enable audio processing (DSP on) in PD.

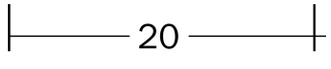
Press the footswitch to start the performance.

Note: the microphone signal should also be directly routed to the speakers so there is an equal balance between the unmanipulated saxophone sound and the sound of the patch.

Improvisation & Notation

Performing this piece requires a basic understanding of tonal improvisation, chord symbols and terminology found in jazz.

This symbol instructs the performer to press and release the footswitch once.



This notation describes this section as lasting approximately twenty seconds

Murmeration

Kier Hall

rec.

Air sound, fluctuating in timbre and volume

20 — 7 — 20

rec.

Add key clicks to air sound. Play faintly-audible chromatic runs.

20 — 15 — 20

rec.

♩ = 60

mf *mf*

20

sporadic tongue slap

p

rec.

fff

2

♩ = 60

f

♩ = 140

Delay effect of semiquavers. Attempt to play in time with patch-controlled rubato.

molto accel. **accel.** **rit.**

fffp *f* *pp* *fff* *mp* *mp* *mf* *p* *f* *p* *ff* *fff* *p* *f*

fpp *f* *pp* *mp* *p* *fff* *ppp* *p* *f* *ppp* *f* *ppp*

mp *f* *p* *mf* *p* *f* *fffp* *fffp* *pp*

f *p* *mf* *cresc.*

rec.

Sounds from the opening slowly return

fff

rec.

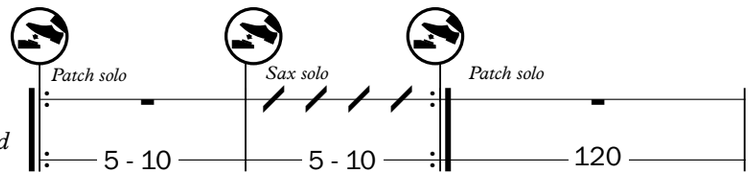
Chord of B♭△ *starts here*

Improvise over chord changes to Donna Lee.
Press pedal to advance to the next chord.
Start slowly like a ballad and build up to a hard-bop frenzy.
The first chord is already sounding.
The section should last approx. four minutes.

B♭△ G7 C7 C-7

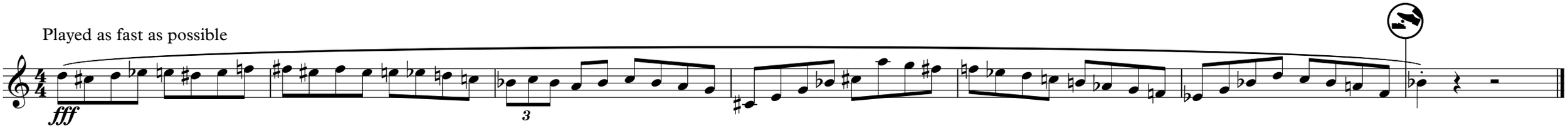
B7(♯11) B♭△ F-7 E7 *etc.* | E♭△ | E♭-7 | B♭△ | G7 | C7 | C-7 | F7 | B♭△ | G7 | C7 | D7 | G-7 | D7 | G- | B♭° | B♭△ | G7 | C-7 | F7 | B♭△ |

Trade short solos with the patch. Play this repeated section a total of eight times.



Play the contents of each cell as many times as you like. Start the section with long, sustained notes and advance to a chaotic frenzy by the end of the piece.

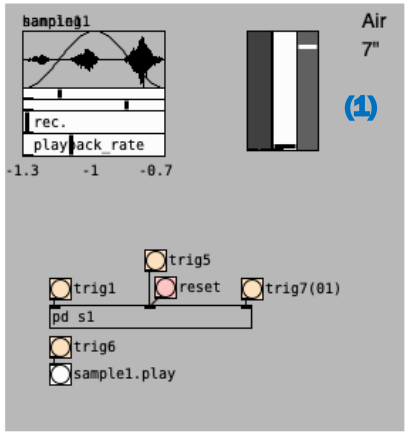
Played as fast as possible

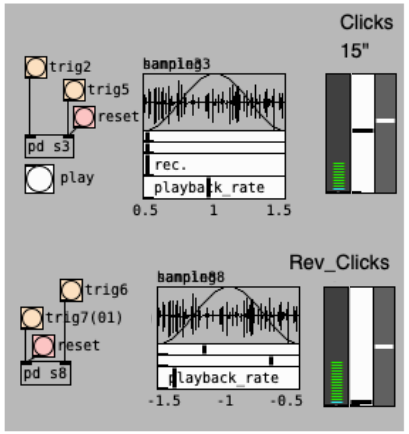


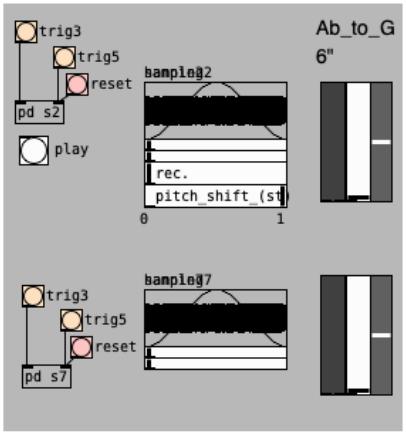
Appendix A: Patch Notes

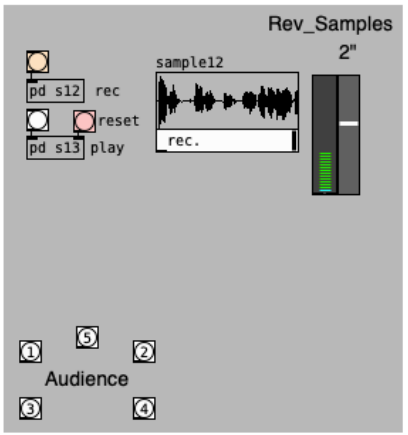
Interface (a)

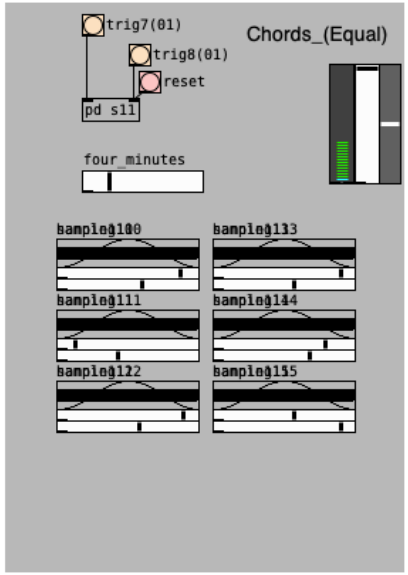
The interface for this patch provides all setup controls and information required to perform this piece. There isn't a requirement for the performer to see this interface but it is helpful for monitoring the patch if an engineer is available for the performance. Each subpatch has its own gain control **(1)** that has been preset to what I believe is a suitable level. However, there may be a requirement to adjust the gain level for different venues depending on their acoustic properties. The performer communicates with the patch via a footswitch MIDI controller. The footswitch makes the patch behave differently depending on when it is pressed during the piece. For example, the first press of the footswitch records a two-second sample of the performer playing air noise while the sixth press instructs the patch to play a sample of the saxophone keys clicking in reverse while varying the playback speed. An engineer operating the patch and following the score will be able to amend any accidental presses by either advancing the footswitch count manually **(2)** or by using subpatch **s6** to choose the exact number of presses from which to resume the performance.

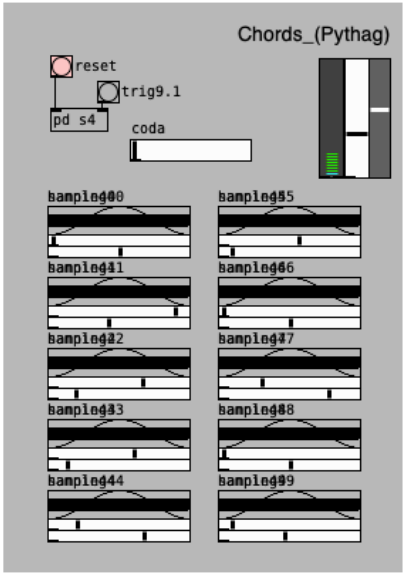


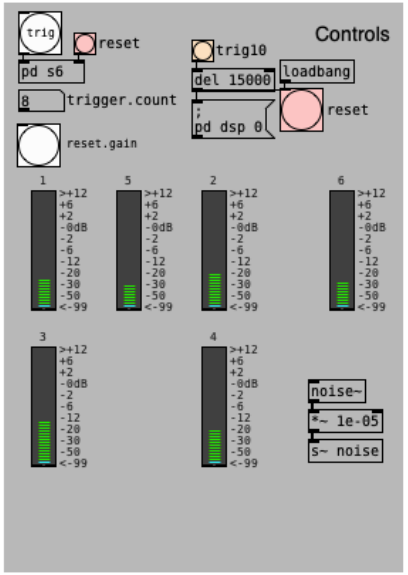


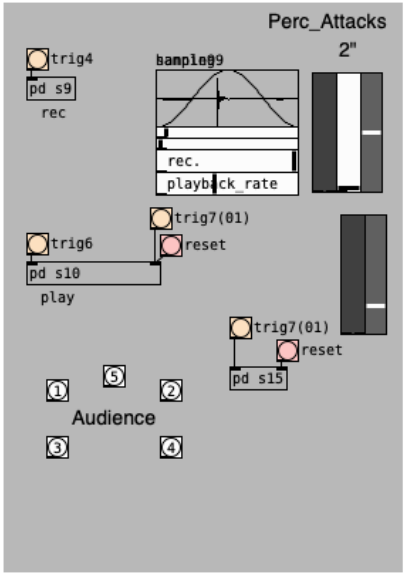




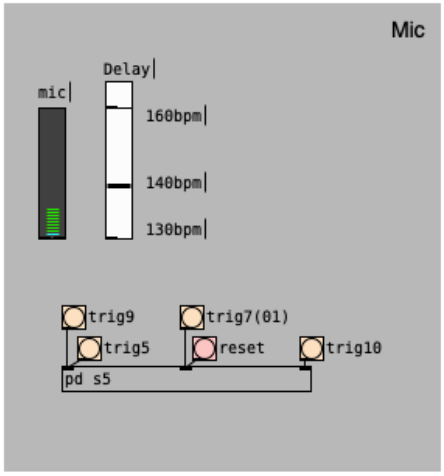


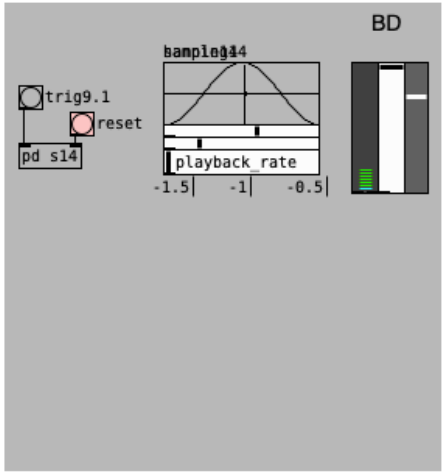


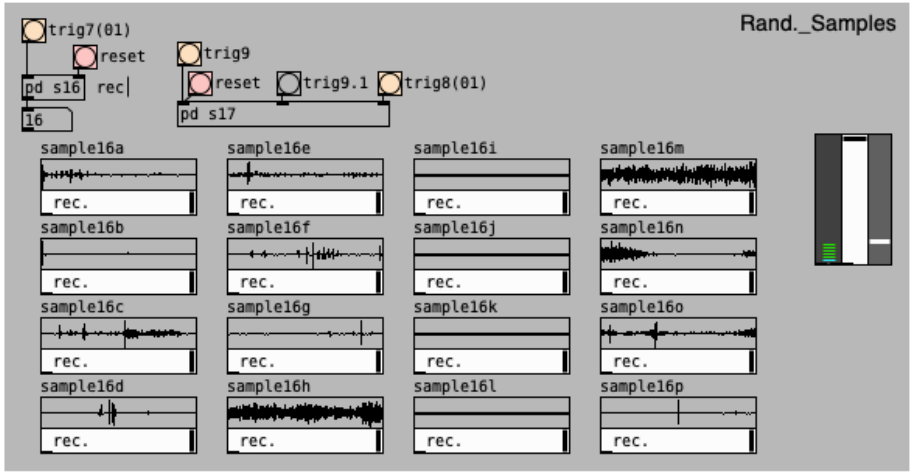


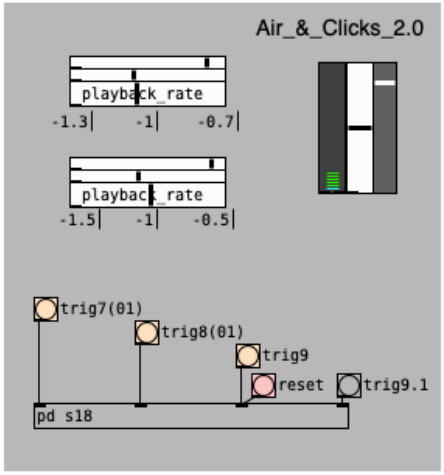


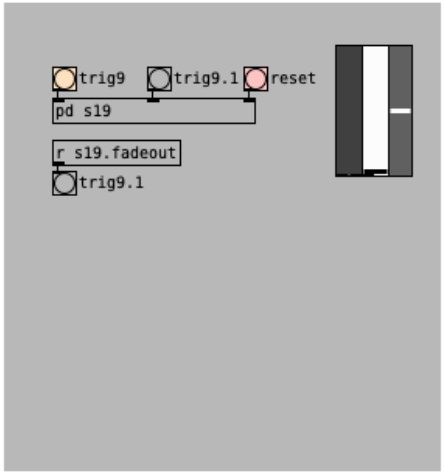
Interface (b)


Mic
mic
Delay
160bpm
140bpm
130bpm
trig9
trig7(01)
trig5
reset
trig10
pd s5


BD
trig9.1
reset
pd s14
playback_rate
-1.5
-1
-0.5

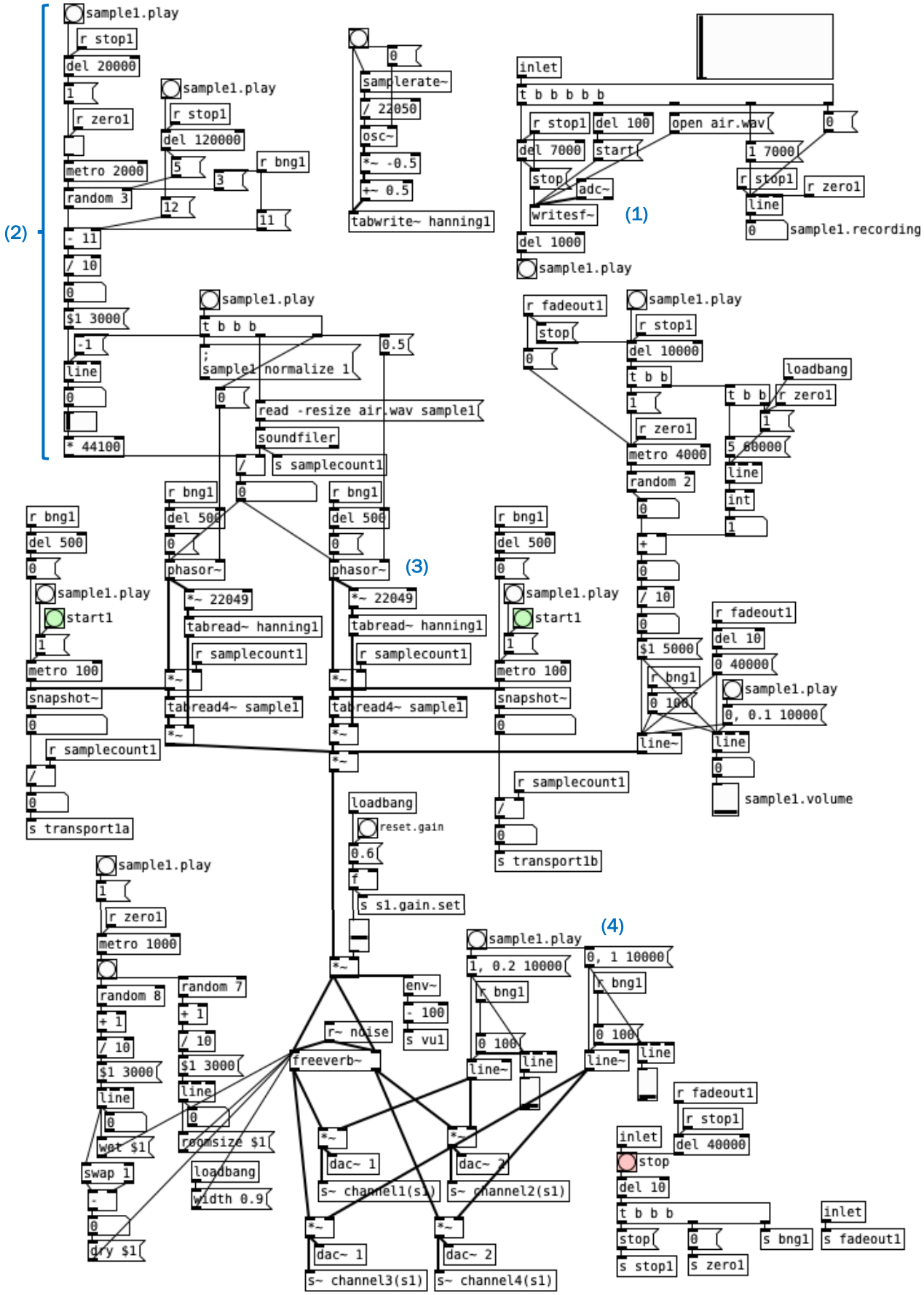

Rand._Samples
trig7(01)
reset
pd s16
rec
16
trig9
reset
trig9.1
trig8(01)
pd s17
sample16a
sample16e
sample16i
sample16m
sample16b
sample16f
sample16j
sample16n
sample16c
sample16g
sample16k
sample16o
sample16d
sample16h
sample16l
sample16p
rec.


Air_&_Clicks_2.0
playback_rate
-1.3
-1
-0.7
playback_rate
-1.5
-1
-0.5
trig7(01)
trig8(01)
trig9
reset
trig9.1
pd s18


trig9
trig9.1
reset
pd s19
r s19.fadeout
trig9.1

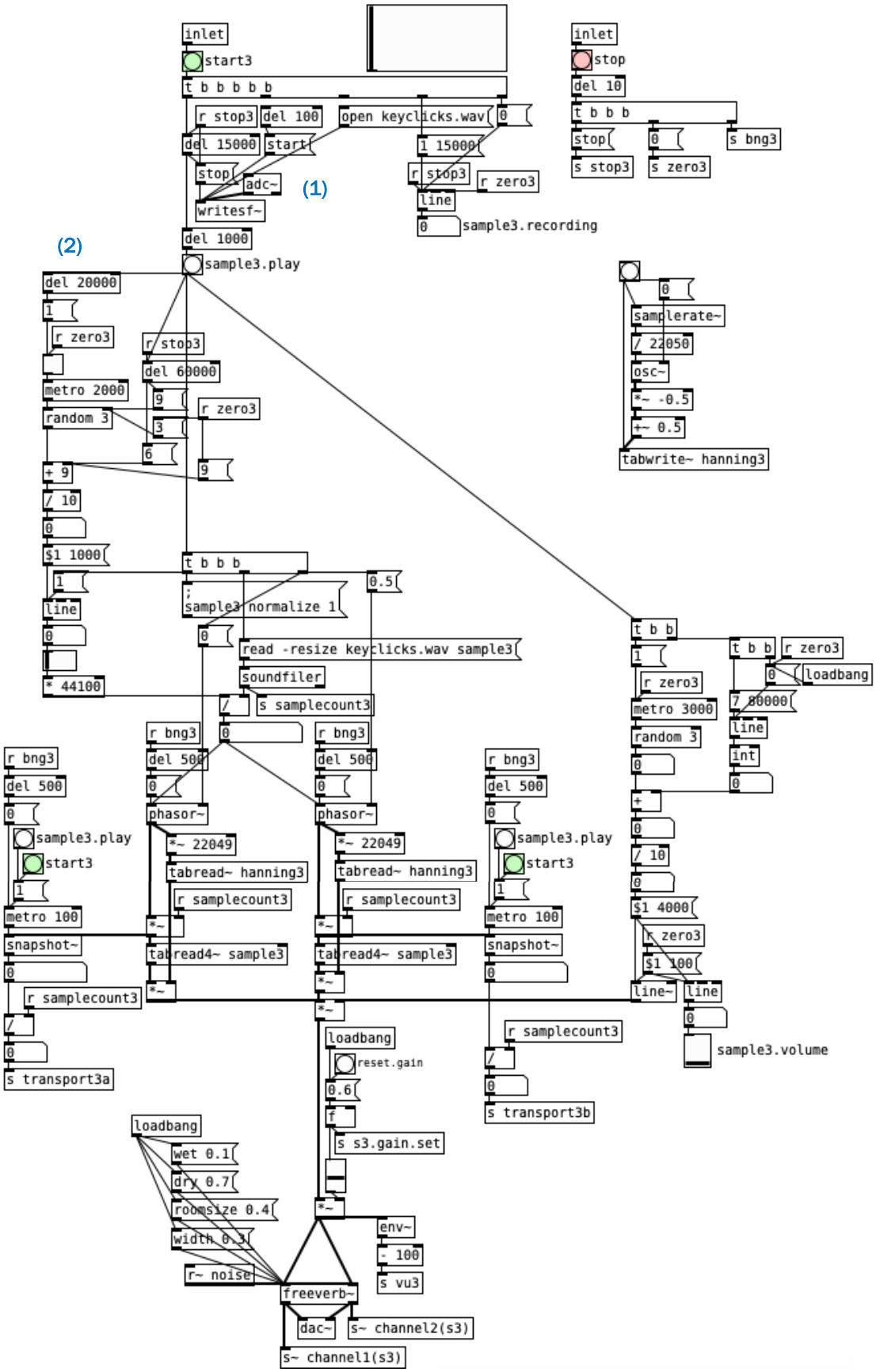
S1

This subpatch, used at the very start of the piece, records a seven-second sample of the saxophonist blowing air through the instrument (1). After the sample is recorded it is played in reverse; the playback speed slowly fluctuating between 0.7 and 1.3 times the original speed (2). The playback of the sample is duplicated and the two waveforms are played half out of phase with one another (3). Combined with a fade in and out at the start and end of each waveform, this process creates a seamless loop that can be played indefinitely. The sound very slowly drifts between the left and right sides of the auditorium (4).



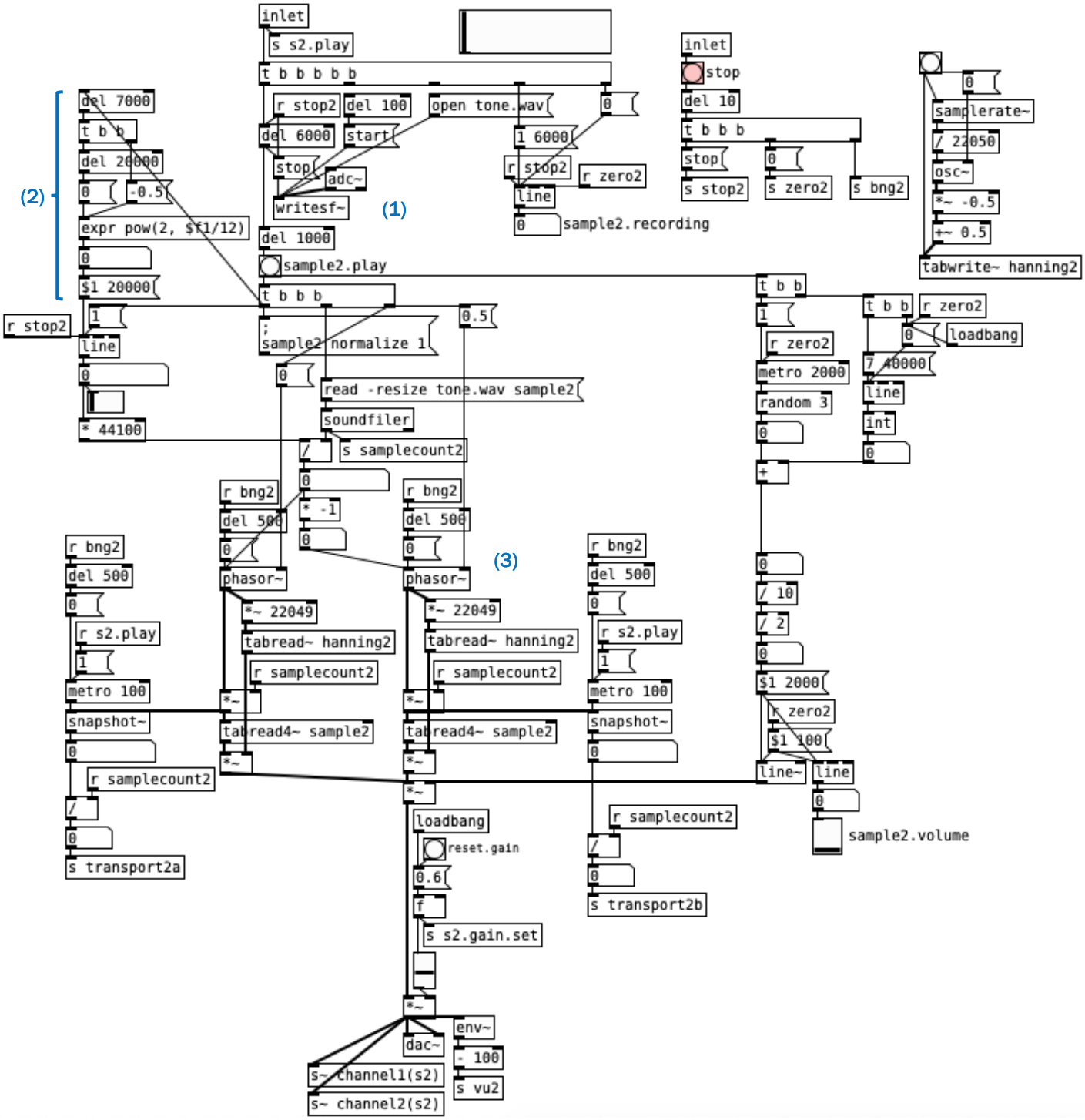
S3

Key clicks are recorded for fifteen seconds (1) and played back in a continuous loop. The playback speed increasingly fluctuates as the section develops (2).



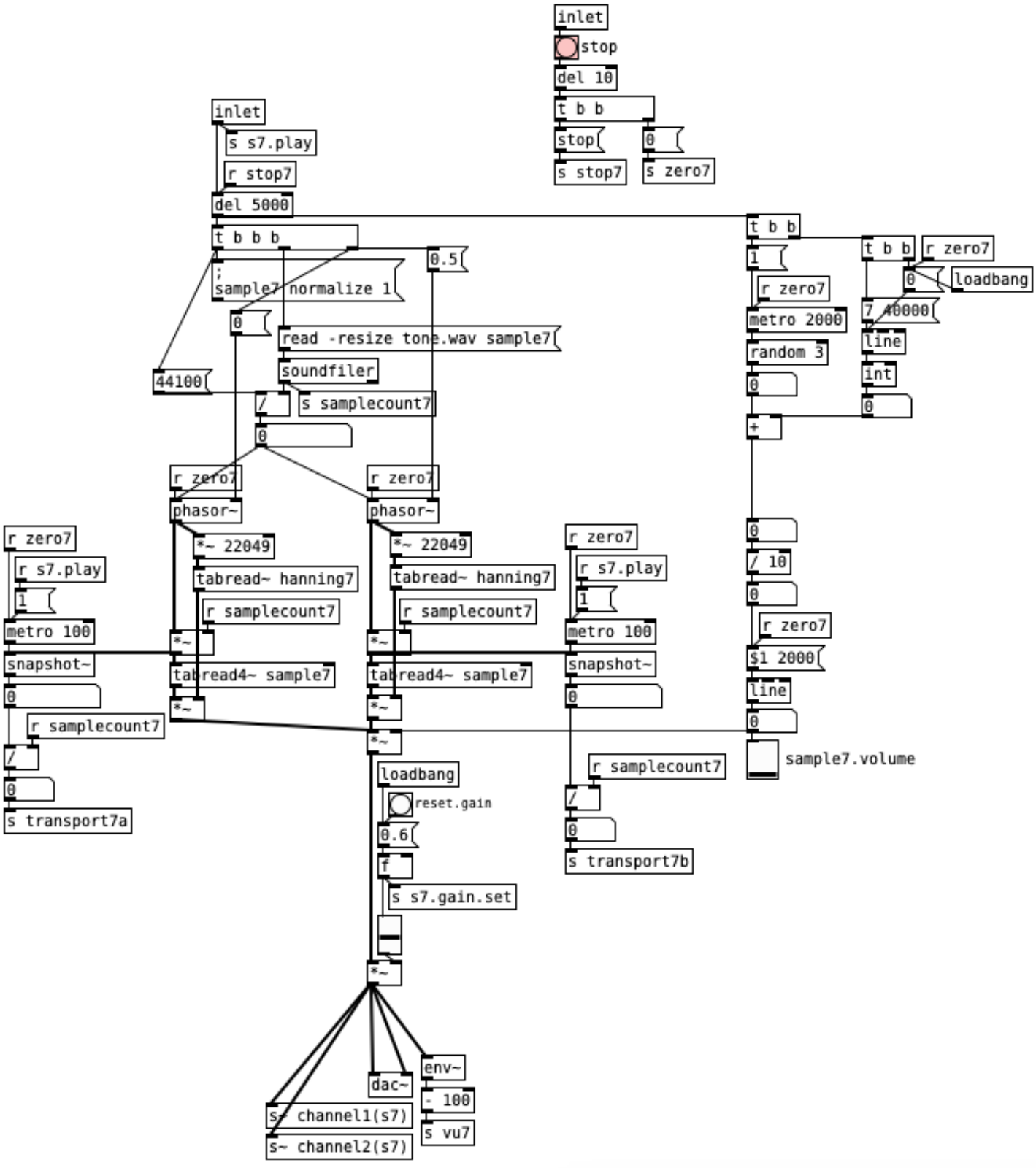
S2

This subpatch records the performer playing a sustained pitch for six seconds (1). Through changing the playback rate (and therefore pitch) of this sample it is used frequently throughout the piece to create a large set of pitches from both equal temperament and three limit Pythagorean tuning systems. In this subpatch the manipulation of the pitch is introduced subtly; the pitch slowly bends down a semitone over twenty seconds while the saxophone remains on the original pitch (2). As with **S1**, the sample is looped through playing a duplicate waveform half out of phase and fading both in and out at the start and end of their respective cycles (3).



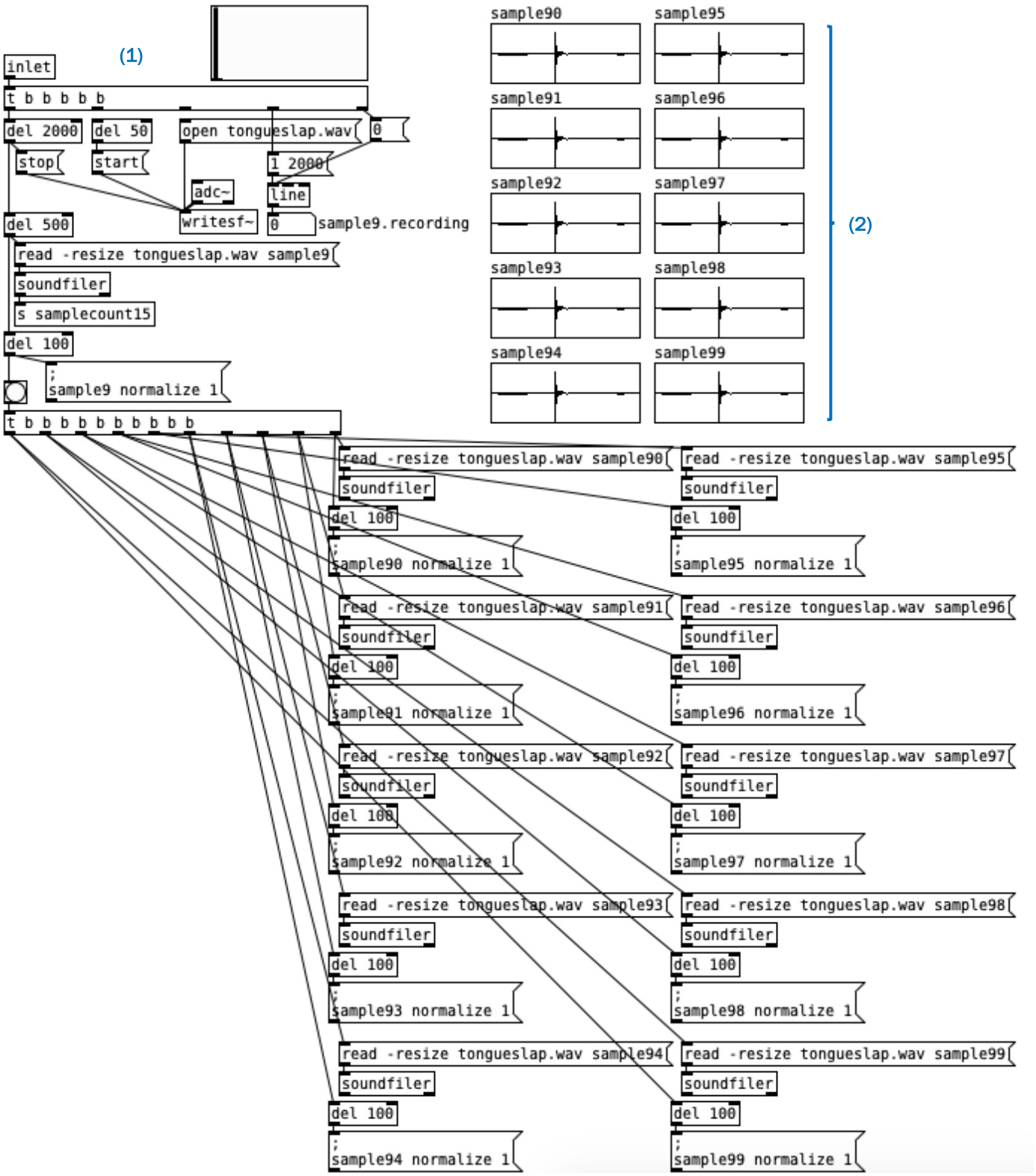
S7

While **S2** is slowly bending the pitch down a semitone this subpatch continues to play the sample at the original pitch, creating harmonic beating that gradually changes speed over twenty seconds.



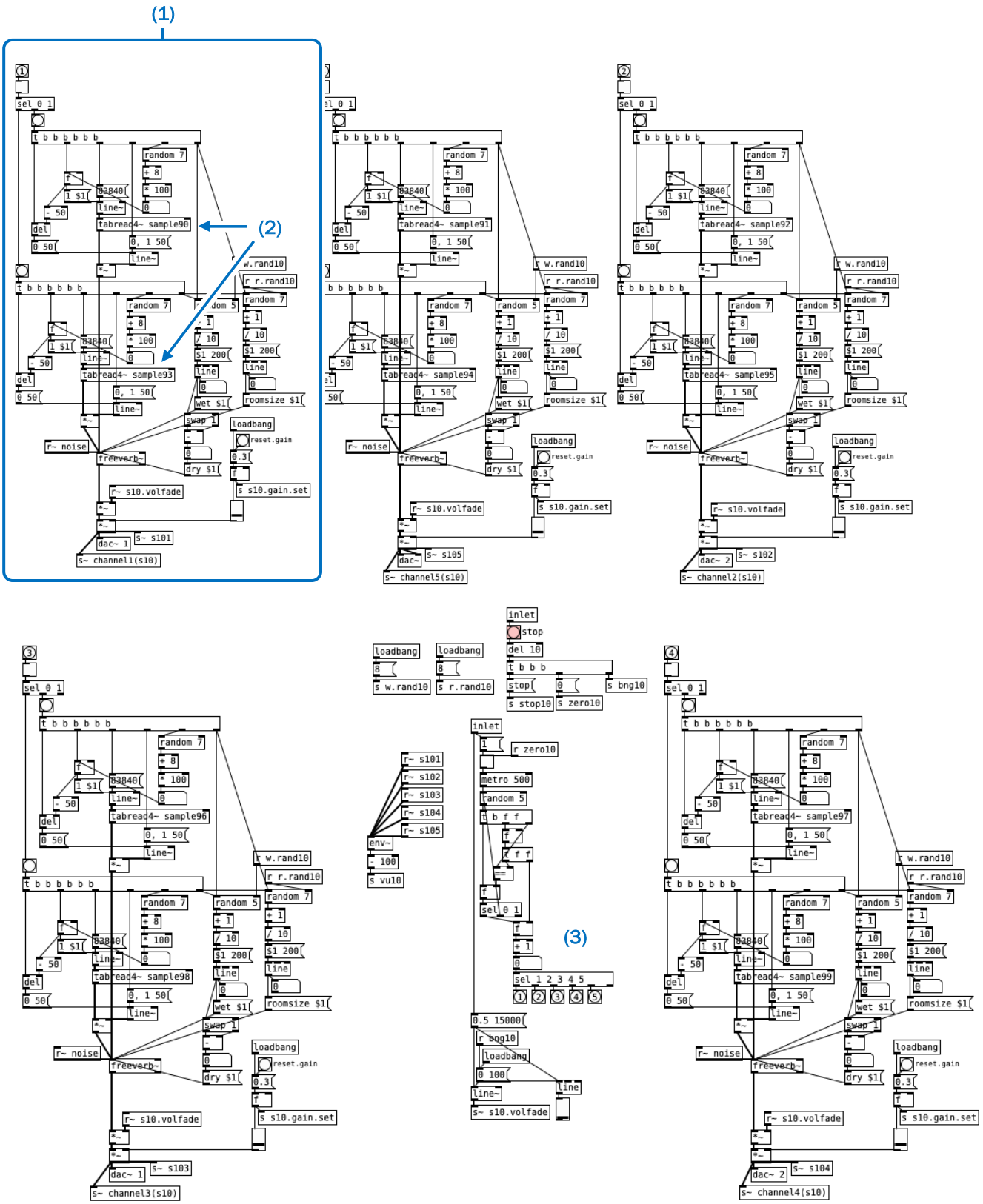
S9

This subpatch records the tongue slap technique (1) and writes the sample to ten identical tables; two for each of the five speakers in the surround sound setup (2). These tables will be read in the following subpatch (**S10**).



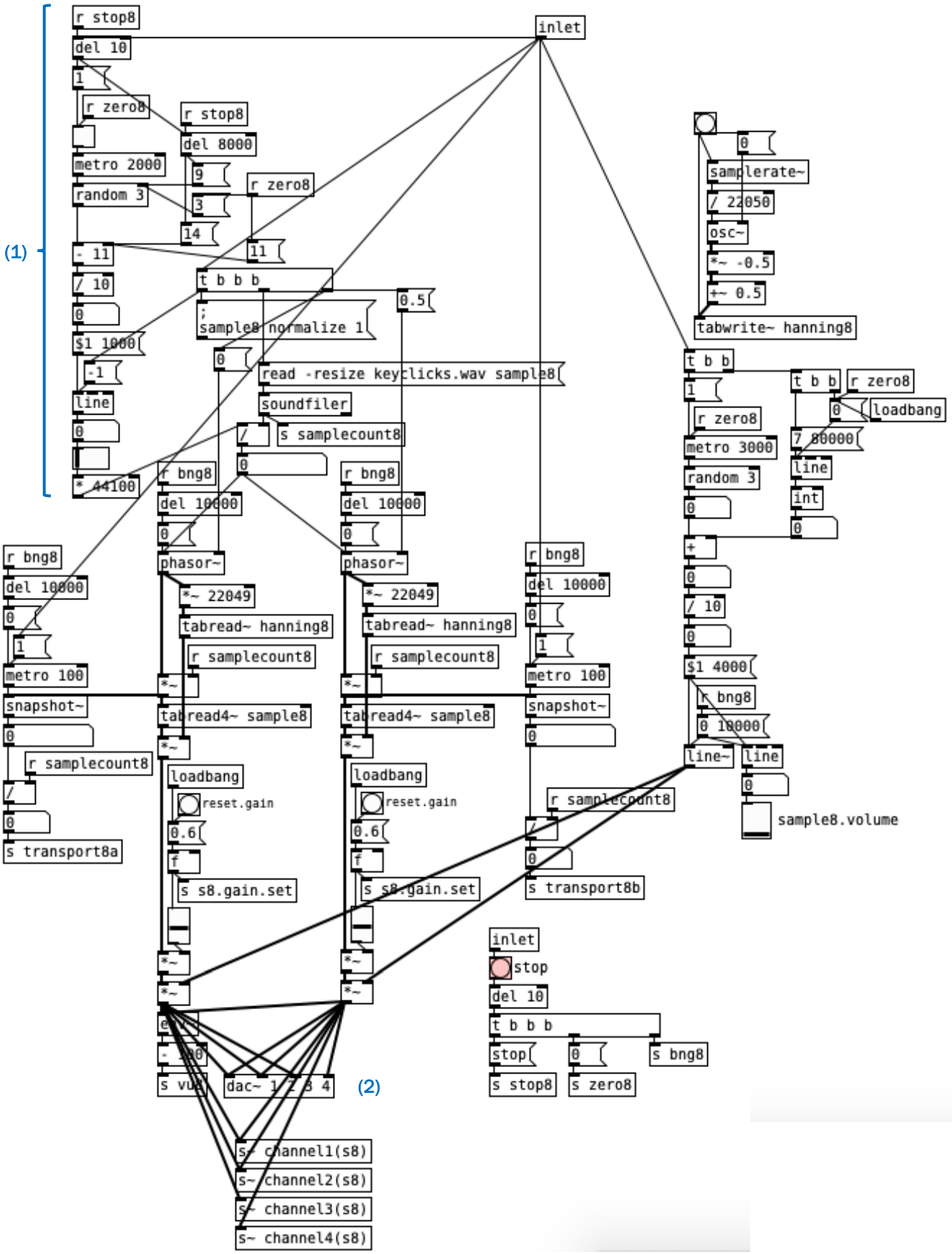
S10

By reading the tables in **S9**, this subpatch allocates two tongue slap samples to each channel (1). For example, the front left speaker uses tables *sample90* and *sample93* (2). The sample being played alternates between the two tables, ensuring the playback doesn't suddenly cut off should the channel replay the sample before the previous playback had finished. The effect of this subpatch is of the tongue slap, played in reverse, being subtly 'bounced' around the audience through random allocation of the speaker within the surround sound setup (3). Playback speed and reverb parameters are randomised, generating different pitches and acoustic effects.



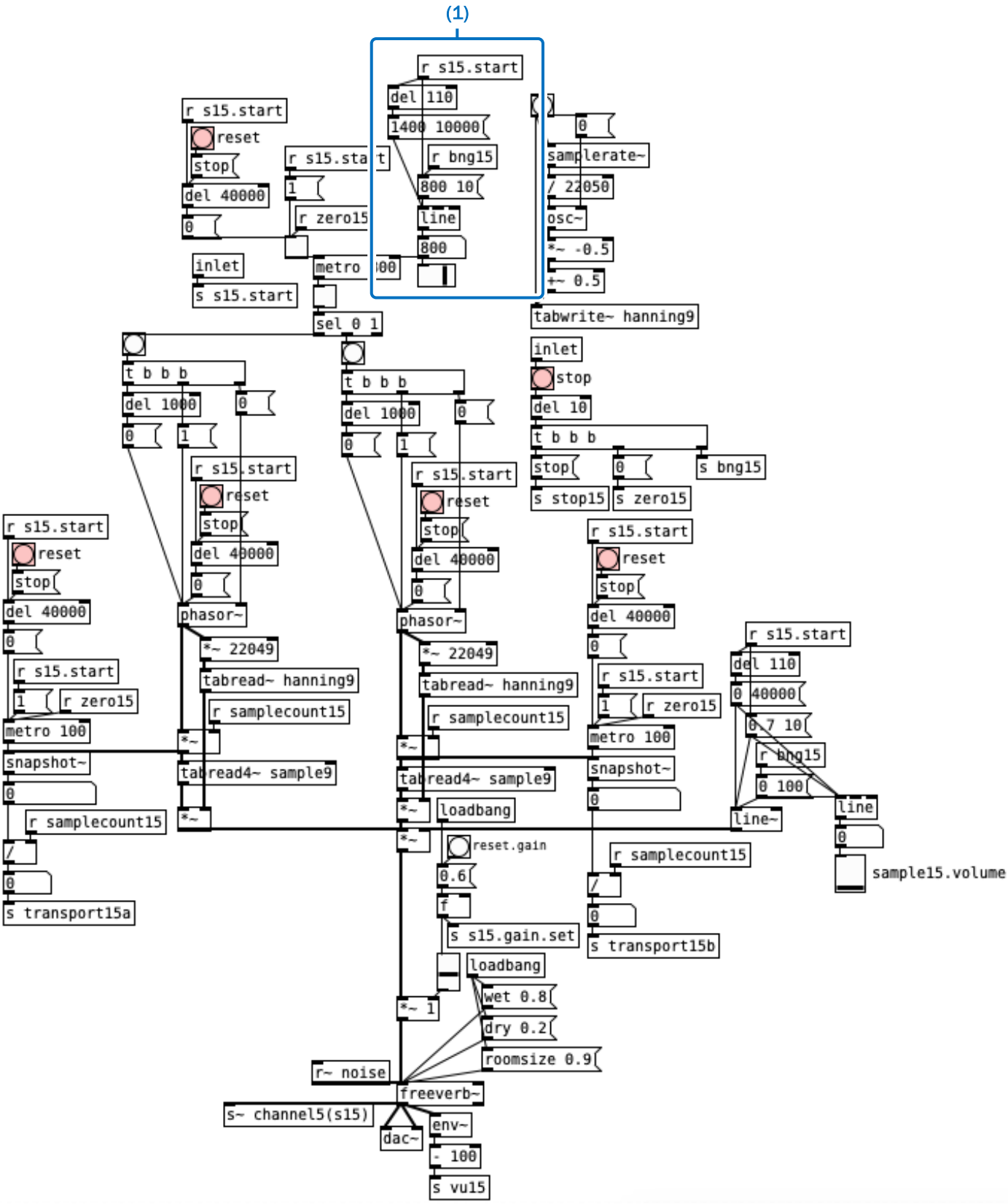
S8

The key clicks sample, having been introduced in **S3**, is transformed further in this subpatch. The sample is reversed and looped, with greater fluctuations in playback speed (1). The sample also drifts between the left and right side of the auditorium (2).



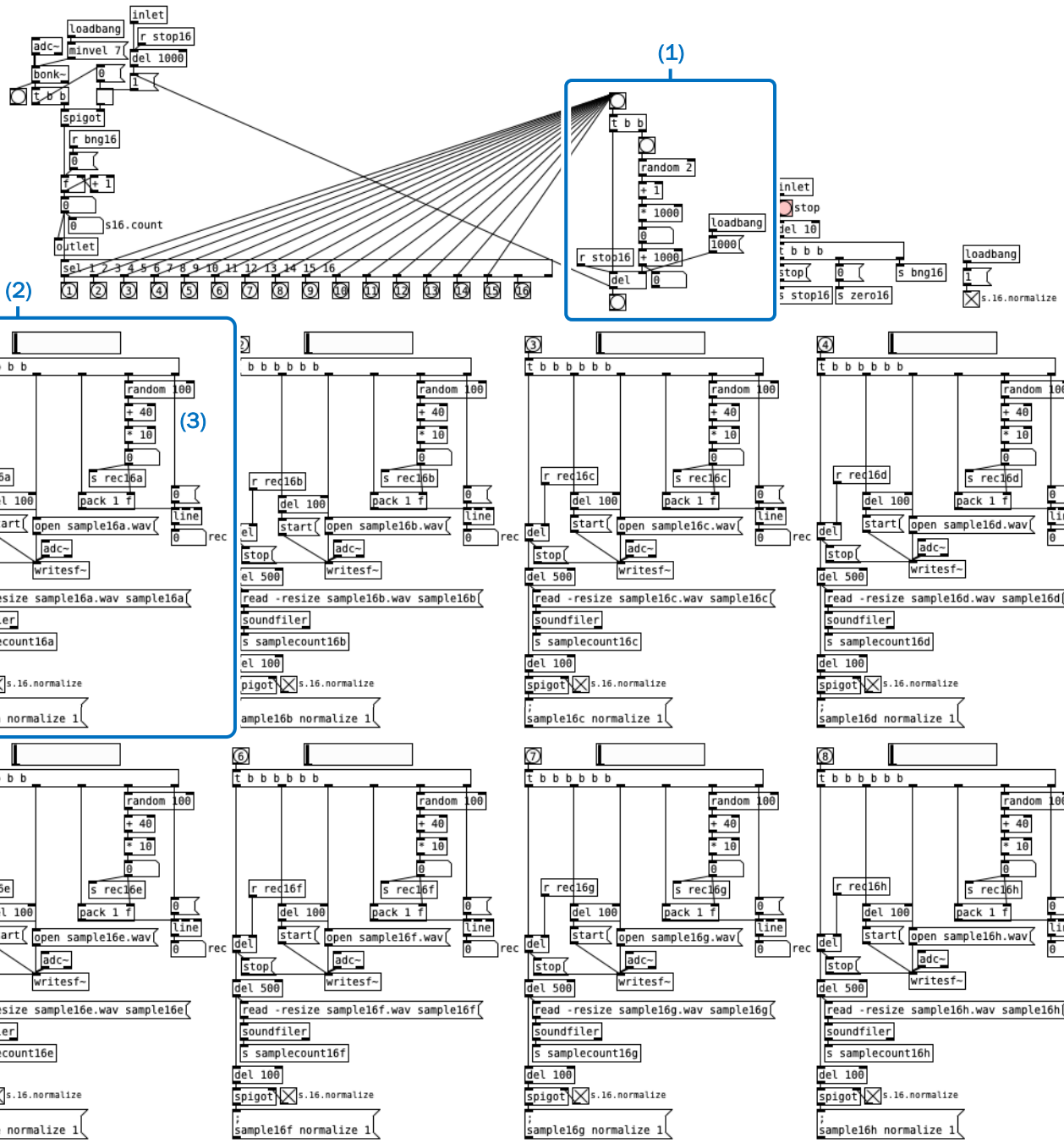
S15

This subpatch is used to bridge the chaotic frenzy of the first section to the ballad-like start of the second. The tongue slap sample played in reverse has a similar timbre to a closed hi-hat often used consistently on off-beats in jazz. This section starts with the reversed tongue slap sample initially played very quickly at a hard-bop tempo before gradually slowing and fading, introducing the ballad theme of the following section (1).

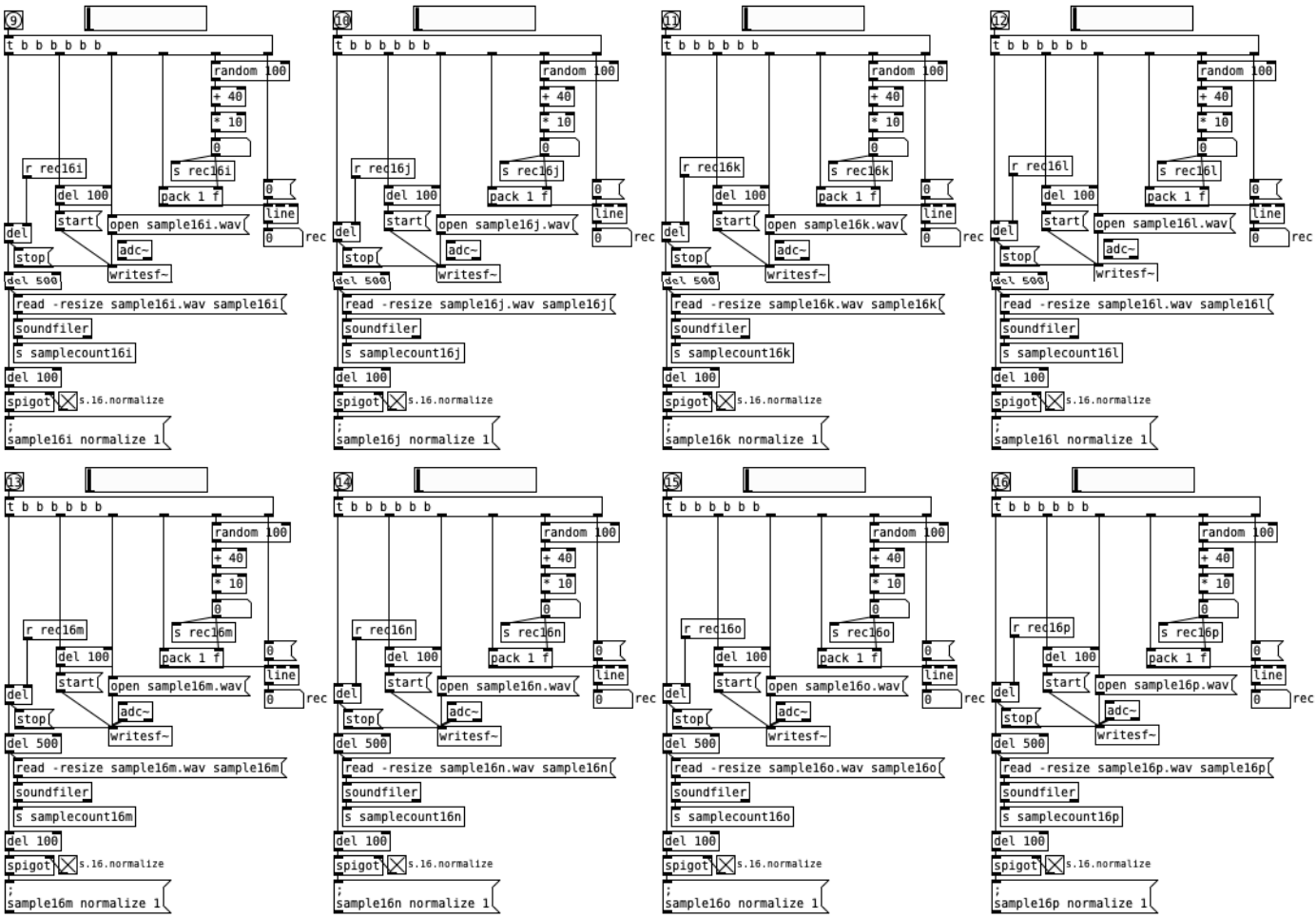


S16

Used throughout the chord-based improvised section, this subpatch automatically records sixteen phrases throughout the section. After a sample is recorded, a debounce time of approximately three seconds ensures all sixteen recording envelopes are not taken from the very start of the section but are instead distributed evenly throughout (1). Each sample is written to a table and stored to be used later in the piece (2). The sample length is randomly selected when the recording starts, ranging from a 400ms to a 1400ms sample (3).

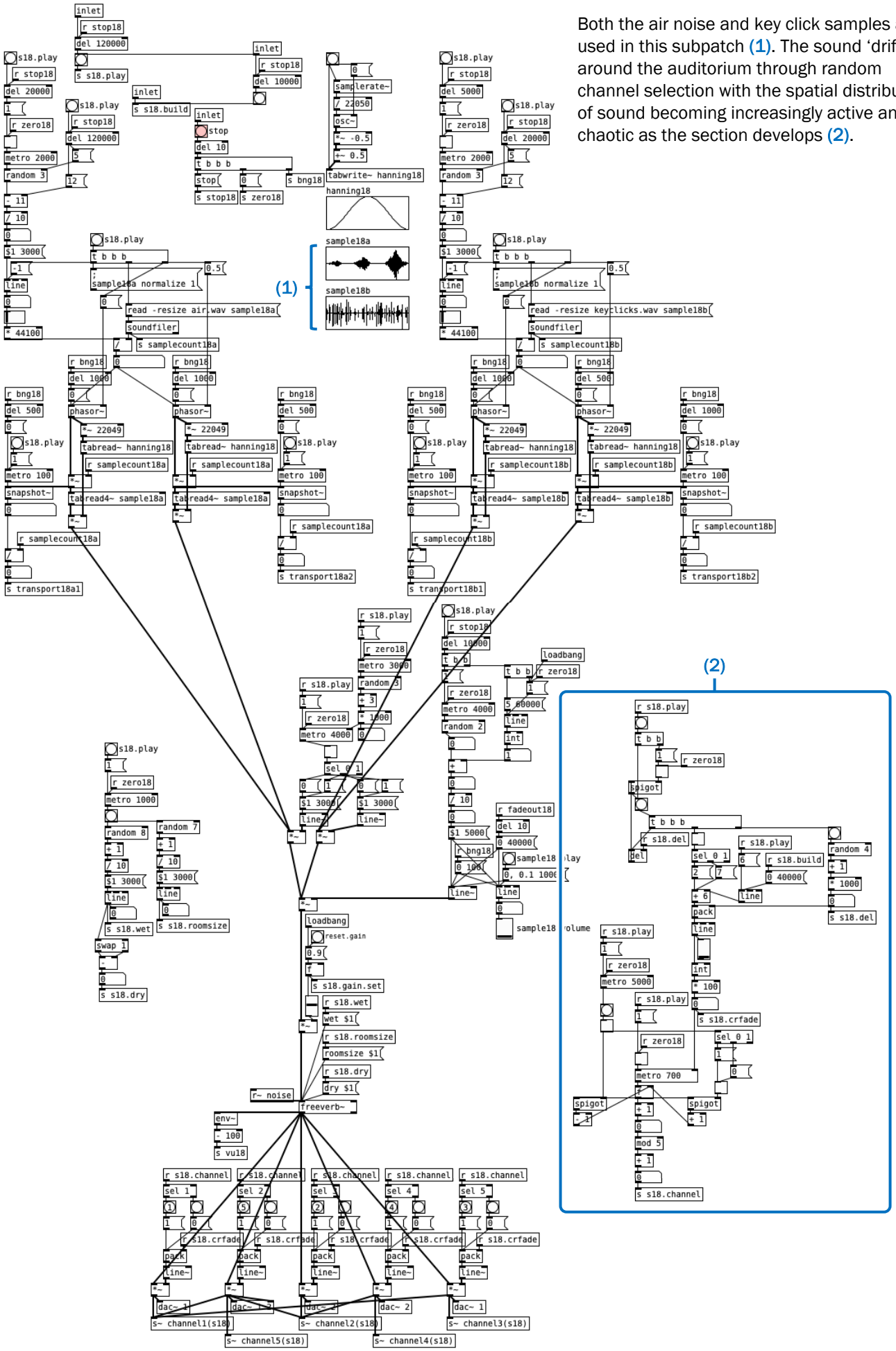


S16 (continued)



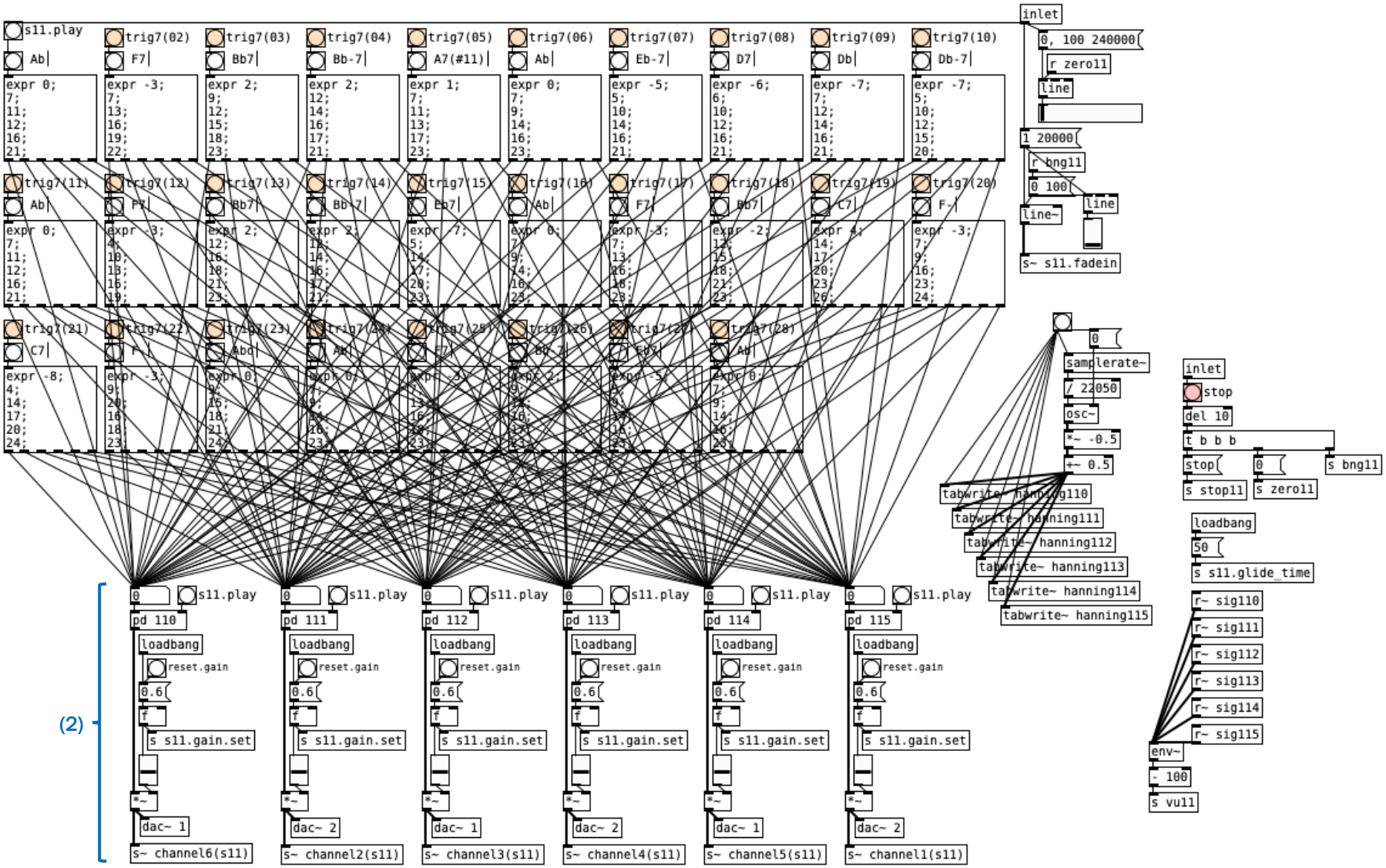
S18

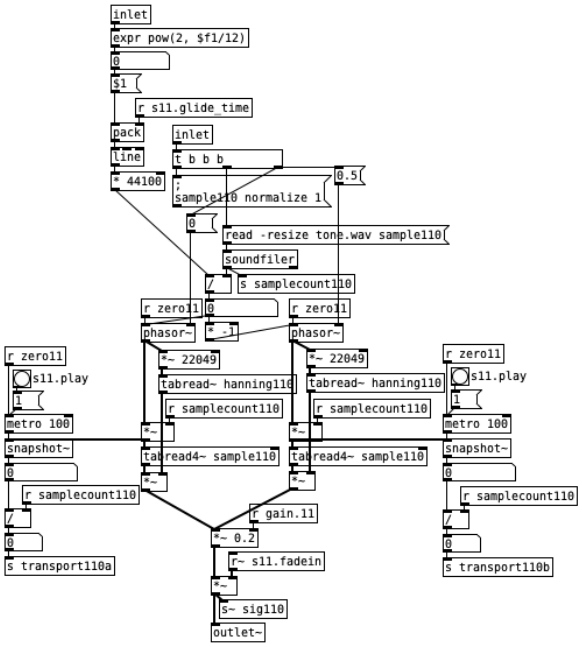
Both the air noise and key click samples are used in this subpatch (1). The sound ‘drifts’ around the auditorium through random channel selection with the spatial distribution of sound becoming increasingly active and chaotic as the section develops (2).

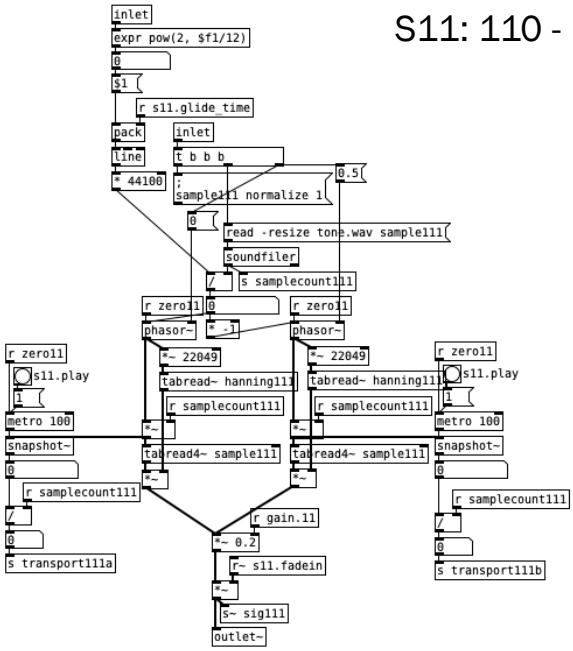


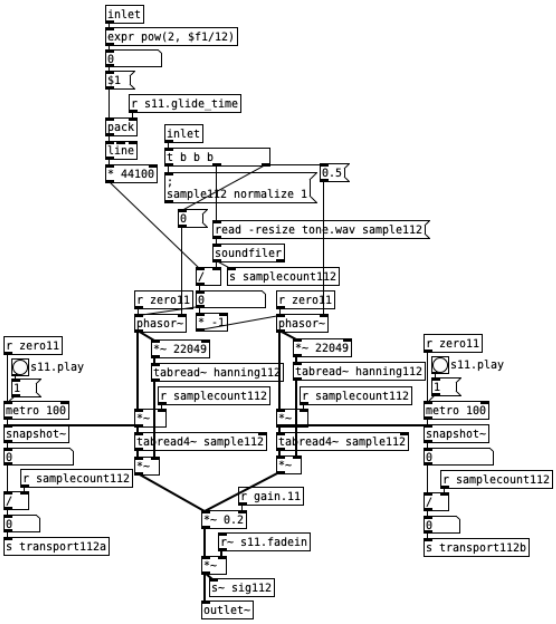
S11

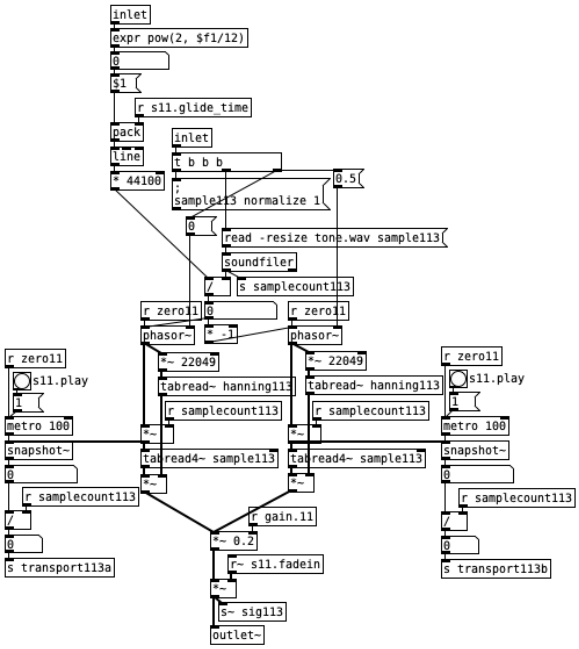
This subpatch is used in the saxophone solo section to advance the chords the performer uses for improvisation. The performer controls the harmonic rhythm in this section through pressing the footswitch to advance the harmony through the chord changes to Donna Lee (1). Each chord has been voiced using five-part harmony with each part being routed to one of the five surround sound channels (2). The intention of designing the texture using this method means an audience member will more clearly hear one of the voices by virtue of being sat nearer one speaker. This will allow an audience member to hear the voice leading of their closest speaker and thus a melody that is more clearly heard by them than other audience members sat elsewhere in the auditorium, creating a unique experience for each member of the audience.

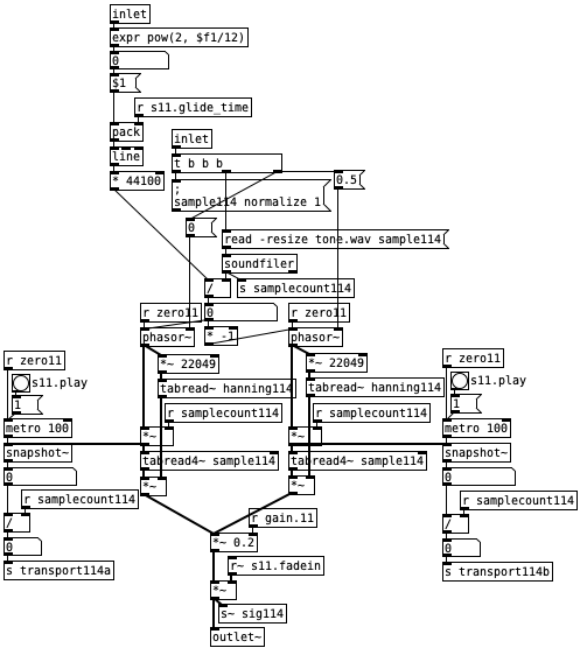


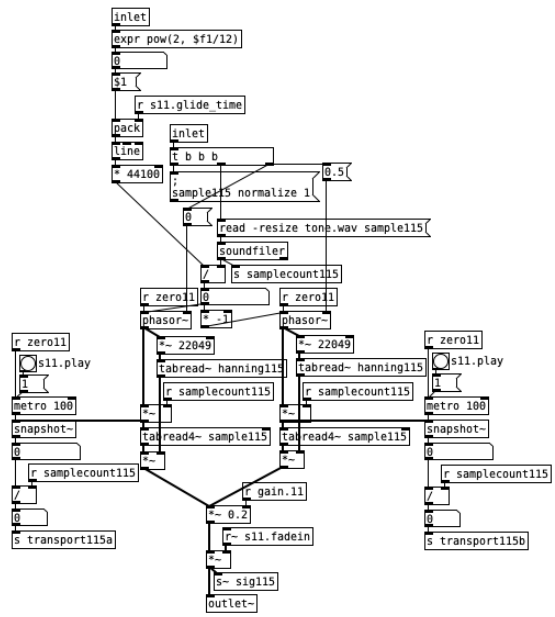

inlet
expr pow(2, $f1/12)
0
$1
r s11.glide_time
pack
line
* 44100
inlet
t b b b
0.5
sample110 normalize 1
0
read -resize tone.wav sample110
soundfiler
/
s samplecount110
r zero11
0
phasor~
* -1
r zero11
phasor~
r zero11
s11.play
1
metro 100
snapshot~
0
r samplecount110
/
0
s transport110a
*~ 22049
tabread~ hanning110
r samplecount110
*~
tabread4~ sample110
*~
r gain.11
*~ 0.2
r~ s11.fadein
*~
s~ sig110
outlet~
r zero11
s11.play
1
metro 100
snapshot~
0
r samplecount110
/
0
s transport110b


inlet
expr pow(2, $f1/12)
0
$1
r s11.glide_time
pack
line
* 44100
inlet
t b b b
0.5
sample111 normalize 1
0
read -resize tone.wav sample111
soundfiler
/
s samplecount111
r zero11
0
phasor~
* -1
r zero11
phasor~
r zero11
s11.play
1
metro 100
snapshot~
0
r samplecount111
/
0
s transport111a
*~ 22049
tabread~ hanning111
r samplecount111
*~
tabread4~ sample111
*~
r gain.11
*~ 0.2
r~ s11.fadein
*~
s~ sig111
outlet~
r zero11
s11.play
1
metro 100
snapshot~
0
r samplecount111
/
0
s transport111b


inlet
expr pow(2, $f1/12)
0
$1
r s11.glide_time
pack
line
* 44100
inlet
t b b b
0.5
sample112 normalize 1
0
read -resize tone.wav sample112
soundfiler
/
s samplecount112
r zero11
0
phasor~
* -1
r zero11
phasor~
r zero11
s11.play
1
metro 100
snapshot~
0
r samplecount112
/
0
s transport112a
*~ 22049
tabread~ hanning112
r samplecount112
*~
tabread4~ sample112
*~
r gain.11
*~ 0.2
r~ s11.fadein
*~
s~ sig112
outlet~
r zero11
s11.play
1
metro 100
snapshot~
0
r samplecount112
/
0
s transport112b

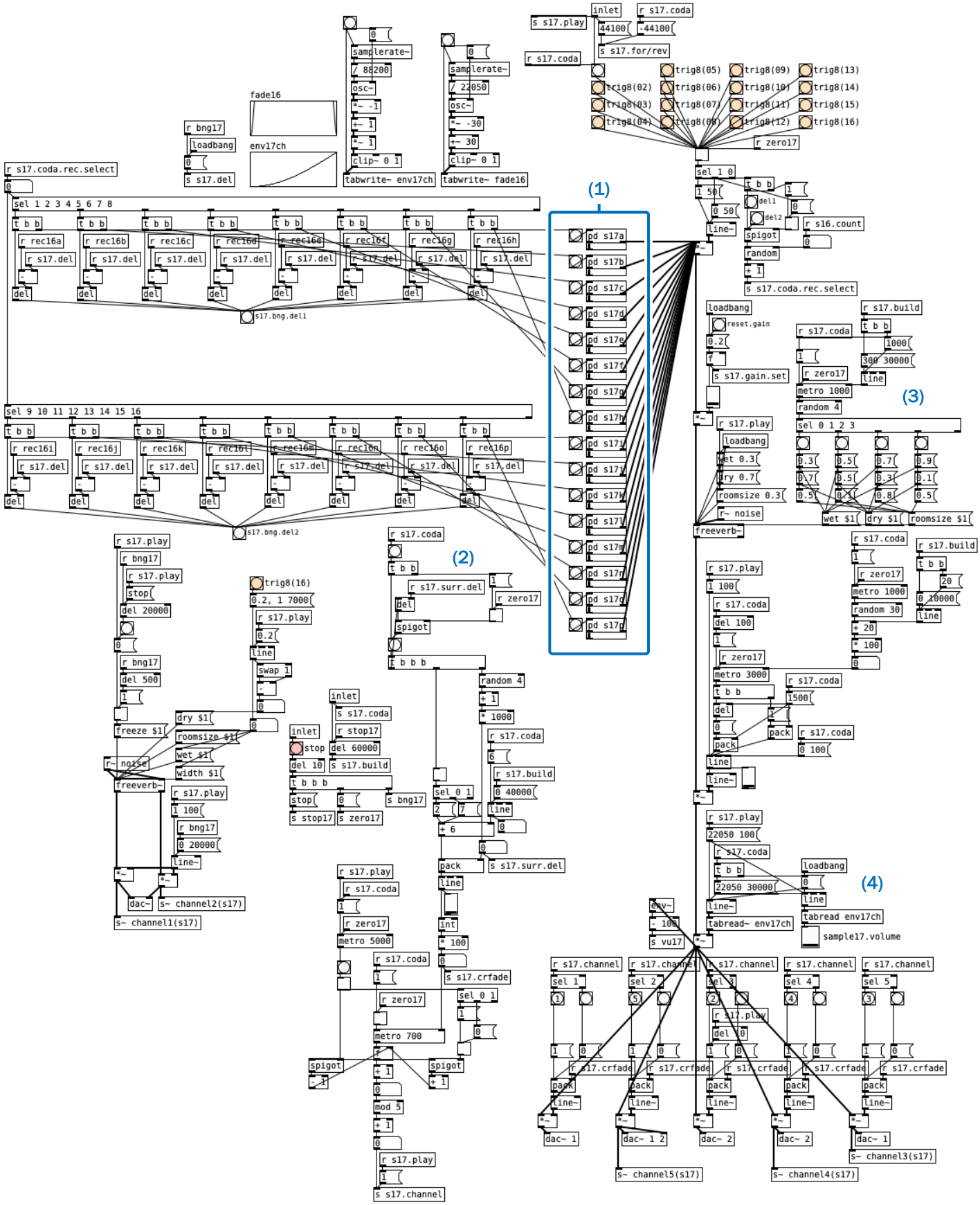

inlet
expr pow(2, $f1/12)
0
$1
r s11.glide_time
pack
line
* 44100
inlet
t b b b
0.5
sample113 normalize 1
0
read -resize tone.wav sample113
soundfiler
/
s samplecount113
r zero11
0
phasor~
* -1
r zero11
phasor~
r zero11
s11.play
1
metro 100
snapshot~
0
r samplecount113
/
0
s transport113a
*~ 22049
tabread~ hanning113
r samplecount113
*~
tabread4~ sample113
*~
r gain.11
*~ 0.2
r~ s11.fadein
*~
s~ sig113
outlet~
r zero11
s11.play
1
metro 100
snapshot~
0
r samplecount113
/
0
s transport113b


inlet
expr pow(2, $f1/12)
0
$1
r s11.glide_time
pack
line
* 44100
inlet
t b b b
0.5
sample114 normalize 1
0
read -resize tone.wav sample114
soundfiler
/
s samplecount114
r zero11
0
phasor~
* -1
r zero11
phasor~
r zero11
s11.play
1
metro 100
snapshot~
0
r samplecount114
/
0
s transport114a
*~ 22049
tabread~ hanning114
r samplecount114
*~
tabread4~ sample114
*~
r gain.11
*~ 0.2
r~ s11.fadein
*~
s~ sig114
outlet~
r zero11
s11.play
1
metro 100
snapshot~
0
r samplecount114
/
0
s transport114b

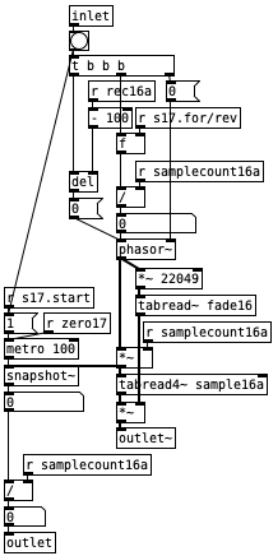

inlet
expr pow(2, $f1/12)
0
$1
r s11.glide_time
pack
line
* 44100
inlet
t b b b
0.5
sample115 normalize 1
0
read -resize tone.wav sample115
soundfiler
/
s samplecount115
r zero11
0
phasor~
* -1
r zero11
phasor~
r zero11
s11.play
1
metro 100
snapshot~
0
r samplecount115
/
0
s transport115a
*~ 22049
tabread~ hanning115
r samplecount115
*~
tabread4~ sample115
*~
r gain.11
*~ 0.2
r~ s11.fadein
*~
s~ sig115
outlet~
r zero11
s11.play
1
metro 100
snapshot~
0
r samplecount115
/
0
s transport115b

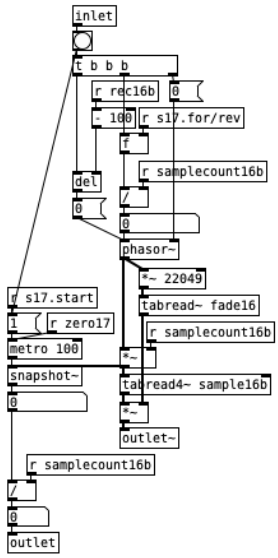
S17

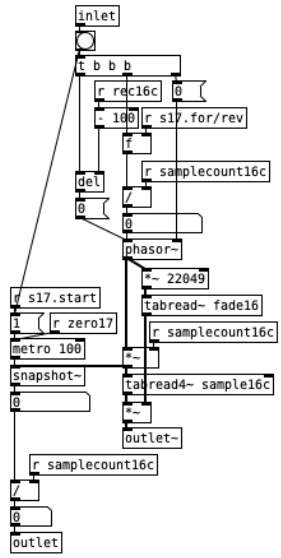
This subpatch is used in the ‘trading’ section found at the top of the score’s second page. Referencing ‘trading fours’ in jazz soloing, the performer and patch trade short solos between one other, each lasting five to ten seconds. The patch generates its solo material by stacking the sixteen samples recorded in **S16** in a random order (1). Any silence that is detected at the start or end of a sample is automatically removed, creating an endless string of phrases. The patch’s solo is controlled by the footswitch, with each press either starting or stopping the improvisation. This subpatch is also used in the coda section, where the samples are played in reverse order with a greater sense of activity through increasingly modulating the phrase length (2), reverb parameters (3) and spatial distribution (4).

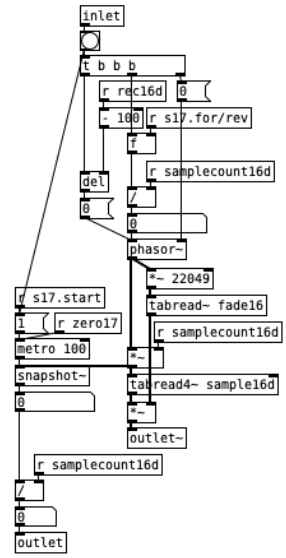


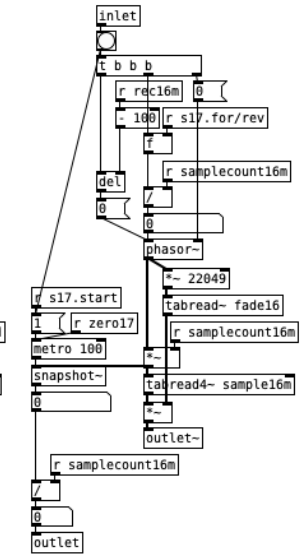
S17: S17a – S17g

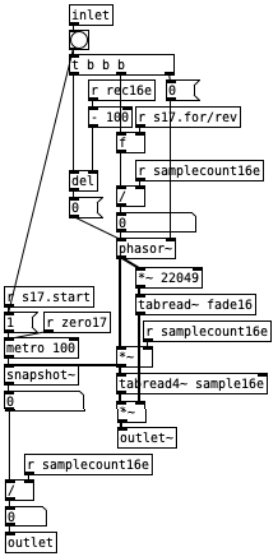


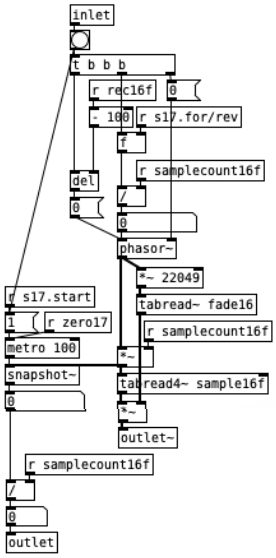


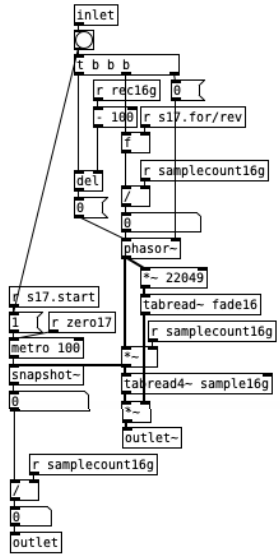


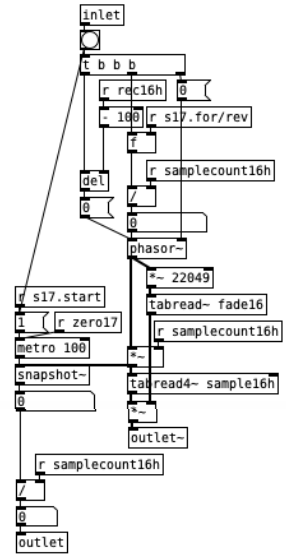


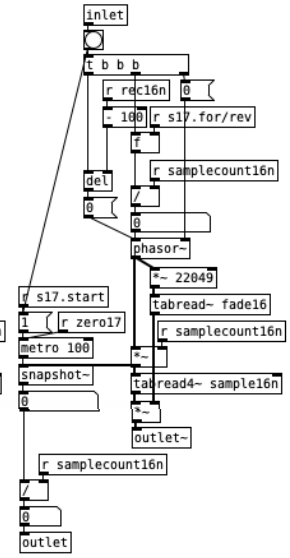


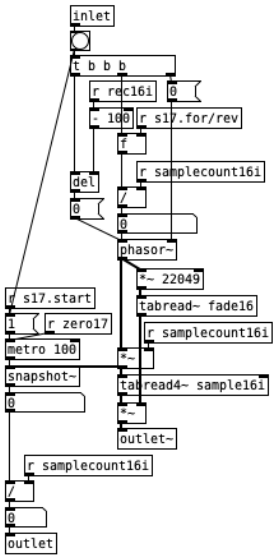


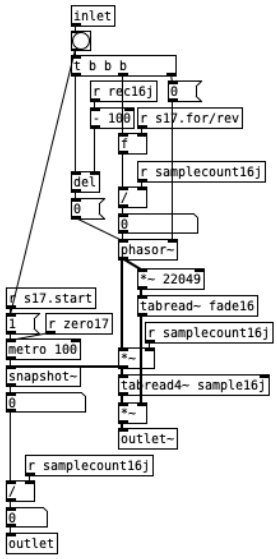


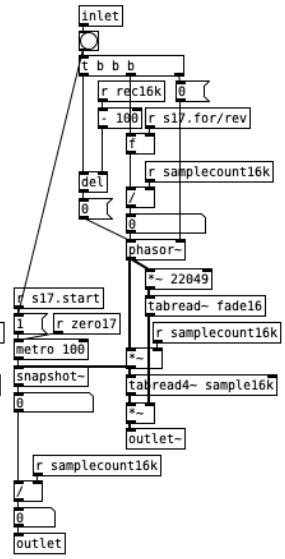


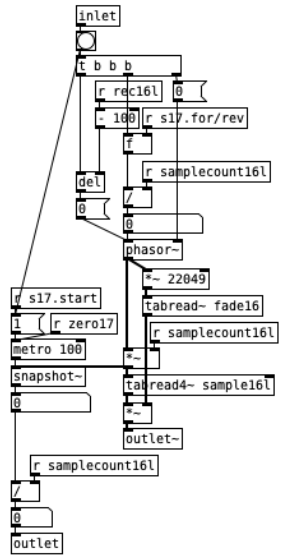


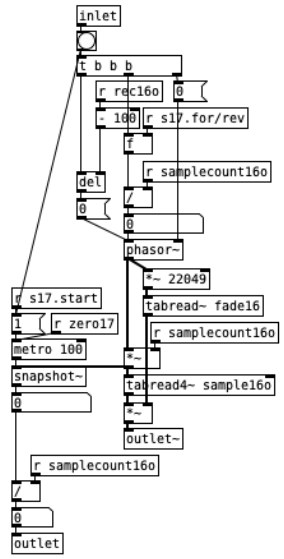






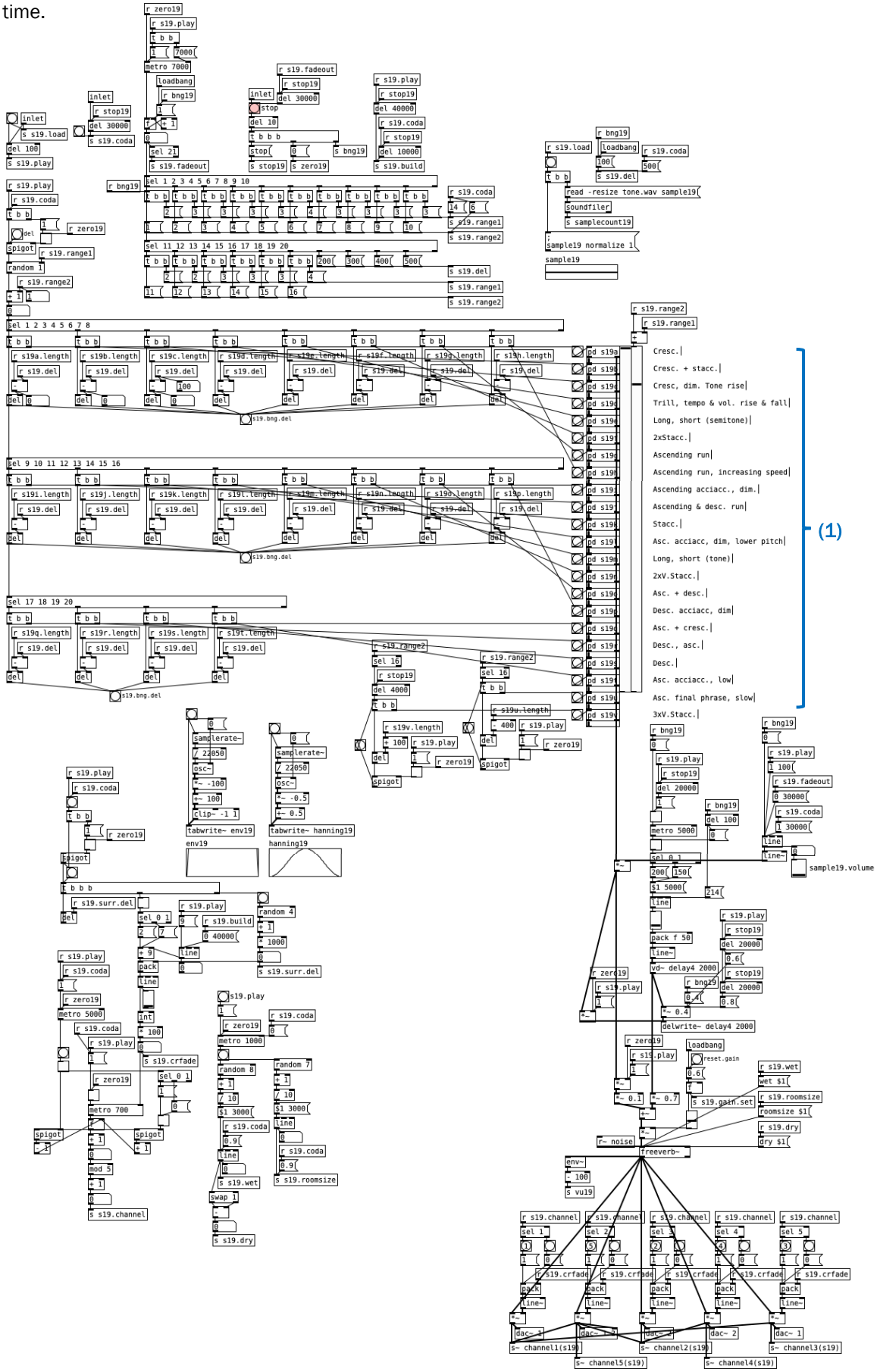


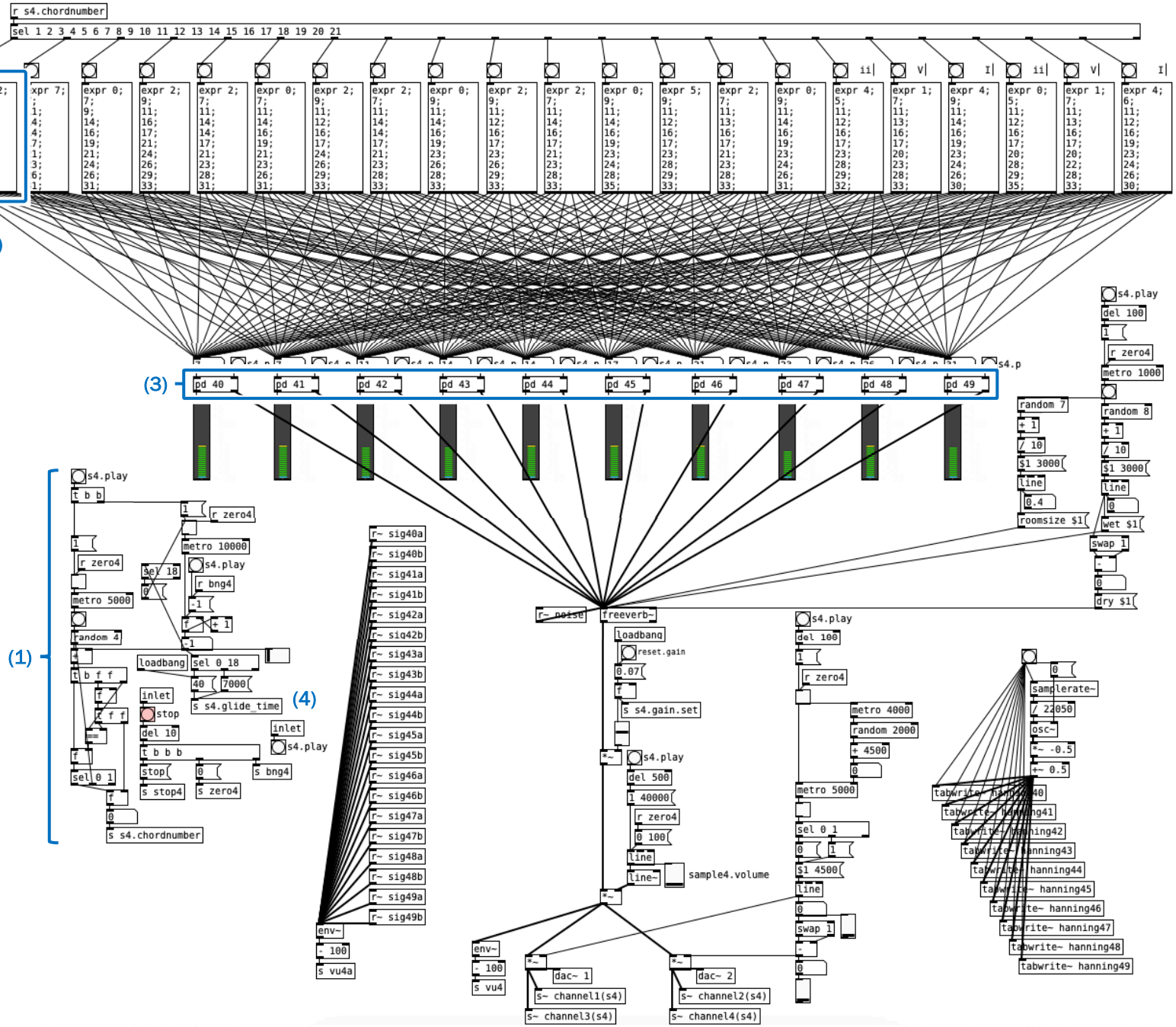




S19

This subpatch is used in the ‘patch solo’ section, immediately after short solos are traded between the performer and patch. Individual phrases have been created with many characteristics such as pitch and exact phrase duration left to randomisation upon performance (1). The material in this subpatch is entirely generated using the sustained tone sample recorded near the start of the piece. Phrases are created by rapidly altering the playback speed and therefore the pitch of the sample. Rhythmic characteristics are created by altering the volume of the sample at the appropriate time.

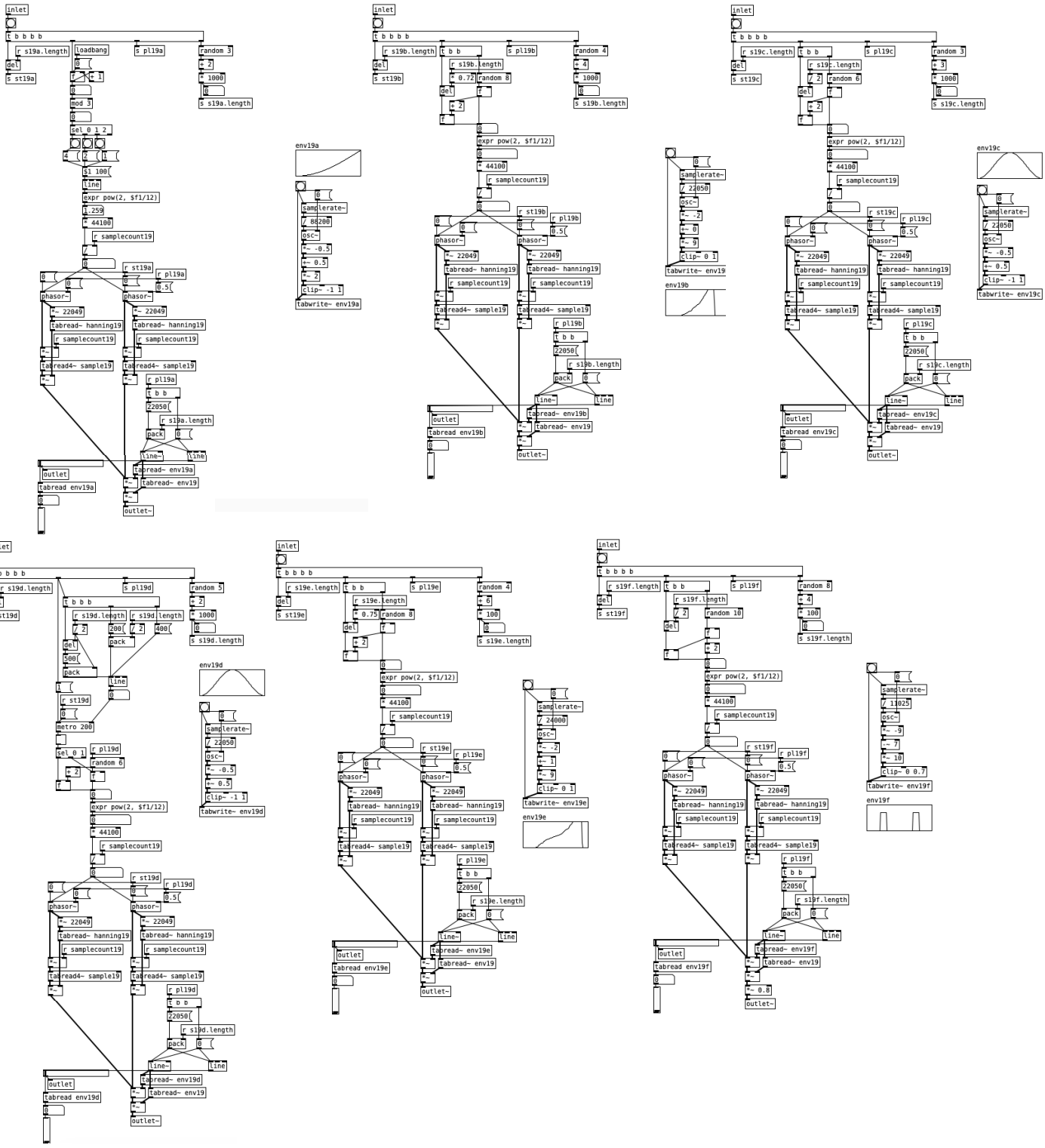




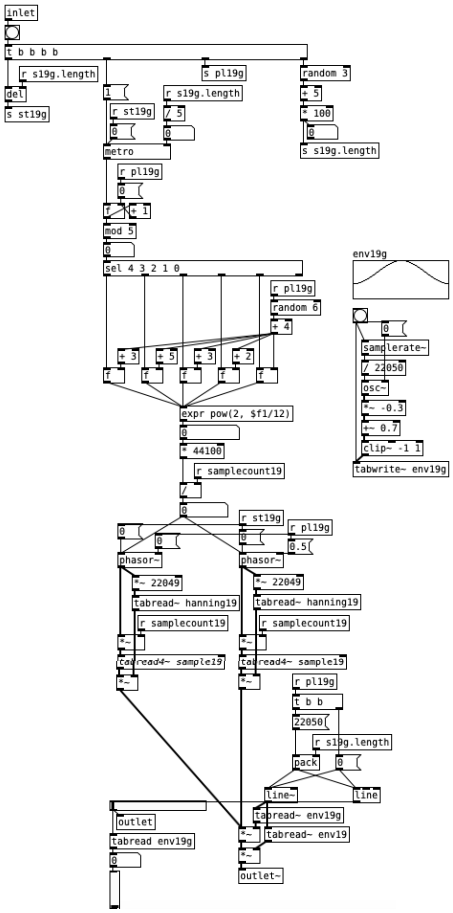
S4

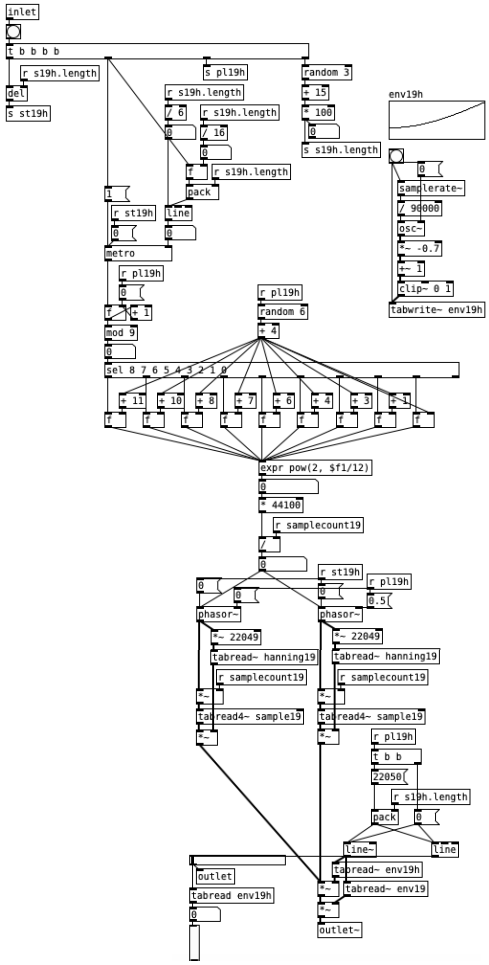
This subpatch governs chords played within the Pythagorean tuning system. The harmony consists of a repeated ii-V-I perfect cadence sequence, become increasingly dissonant through developing the voicings as the section advances. Harmonic rhythm is controlled by the patch (1) with each box (2) describing a chord voicing, indicated via the number of semitones from the sampled A♭ taken at the start of the piece. The ten subpatches (3) each interpret one of the ten numbers (2) and convert them into pitches by adjusting the playback rate. Toward the end of the section the transition from one pitch to the next becomes a seven-second portamento, only allowing the destination pitch to be heard very briefly before it begins to change again (4). This process creates pitch turbulence as the piece nears its end, mimicking the increasingly frantic saxophone playing and other patch components.

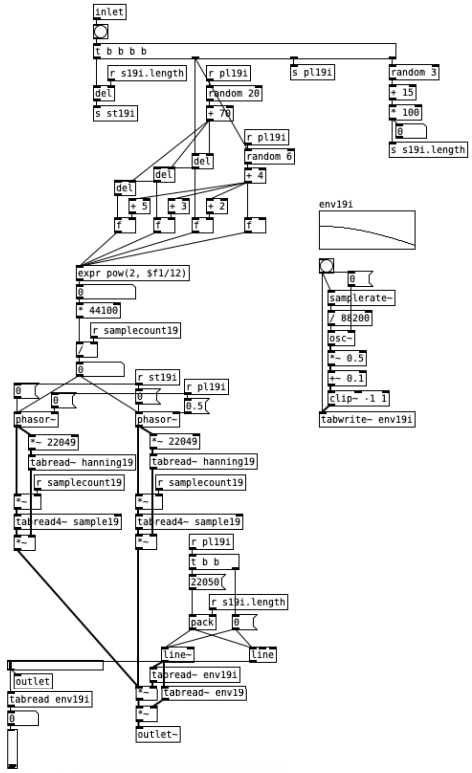
S19: S19a – S19f

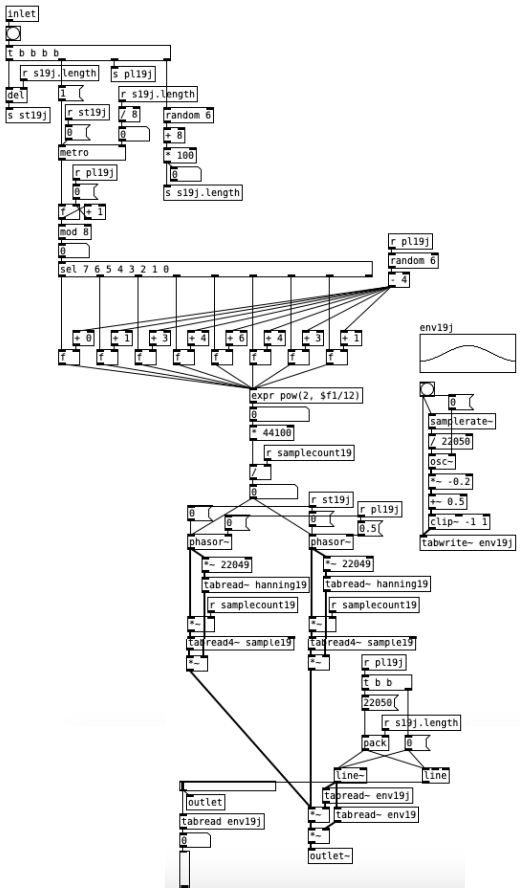


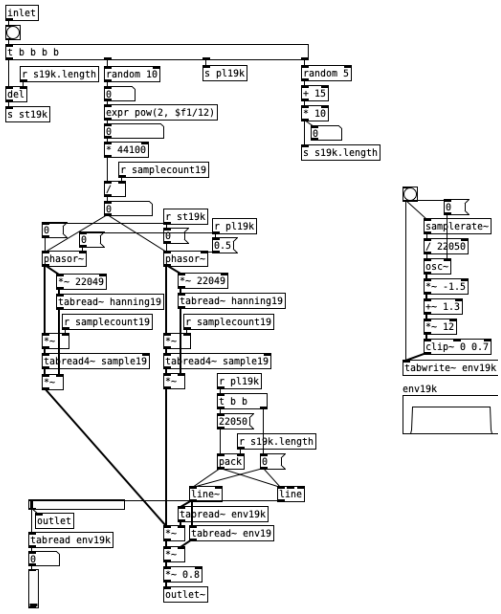
S19: S19g – S19l

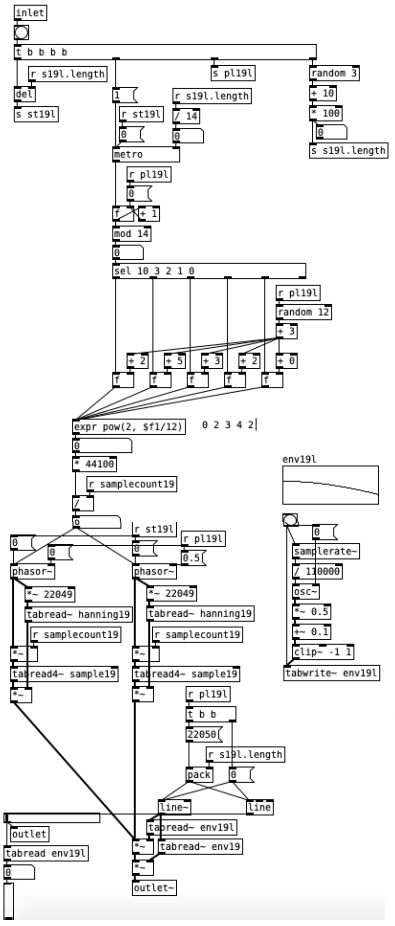




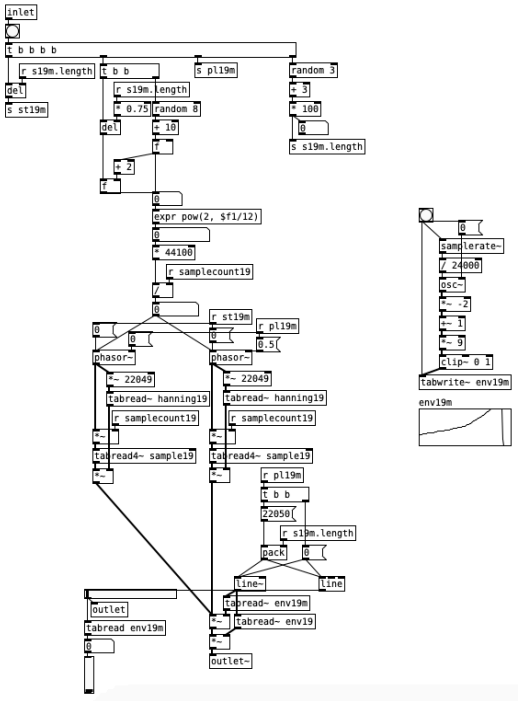


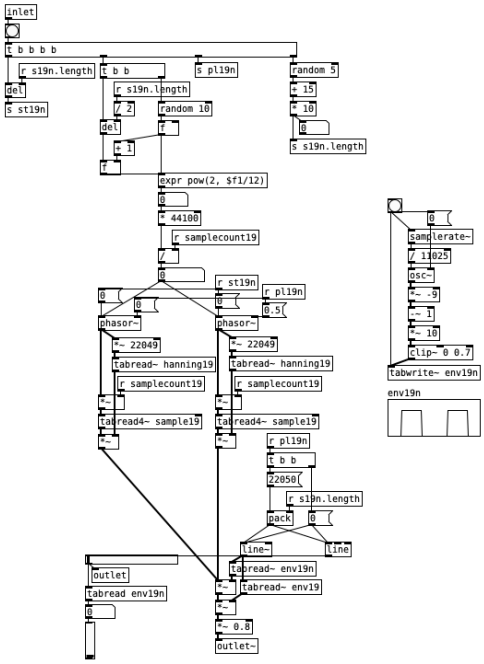


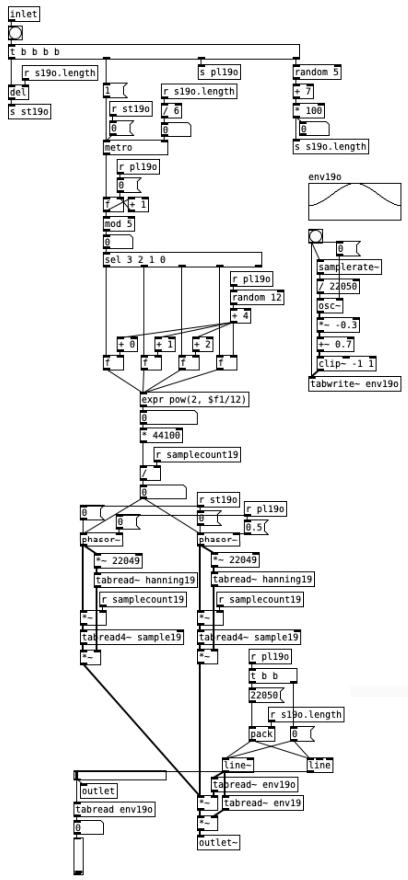


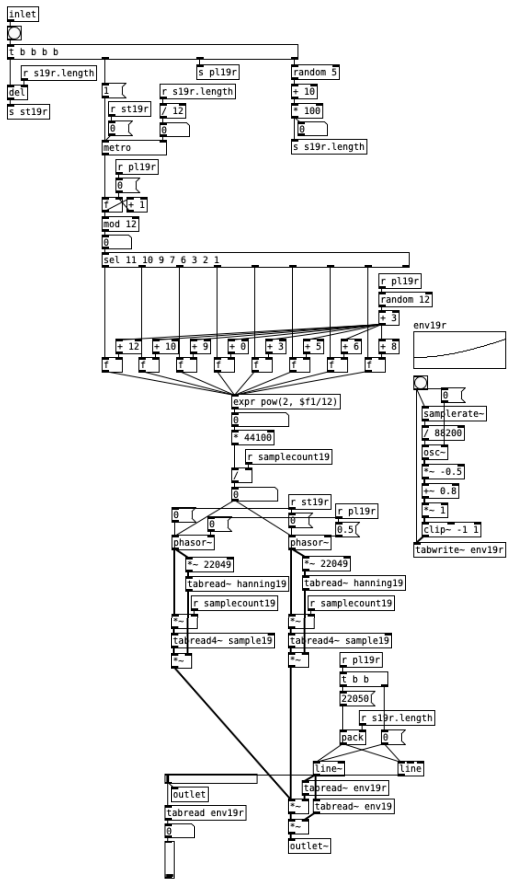


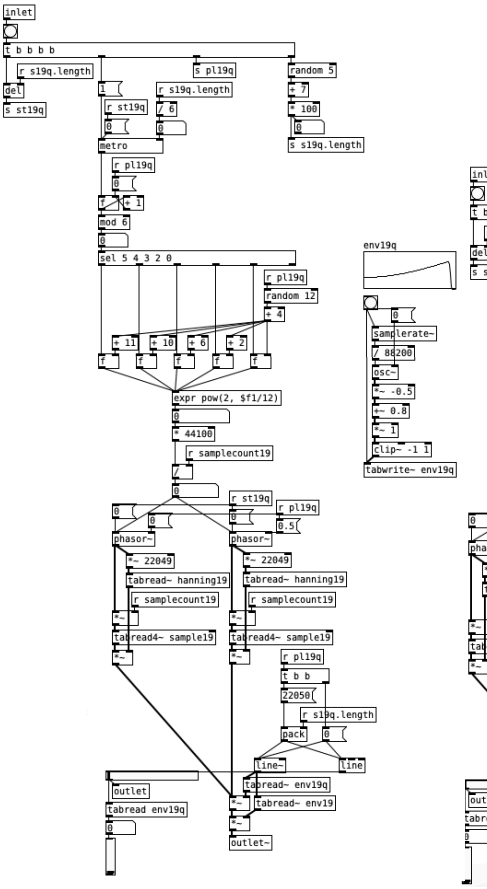
S19: S19m – S19r

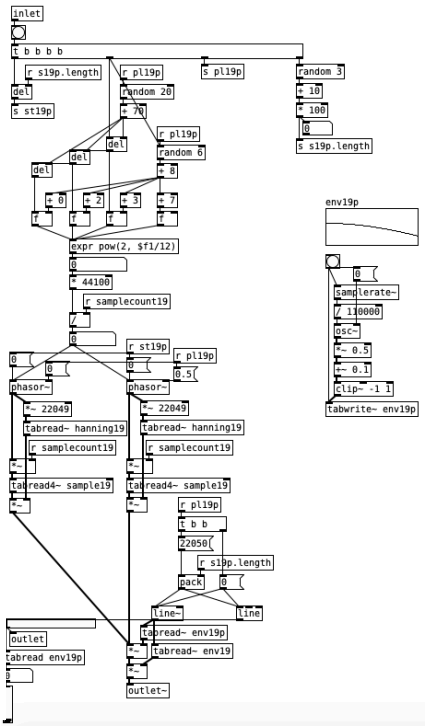


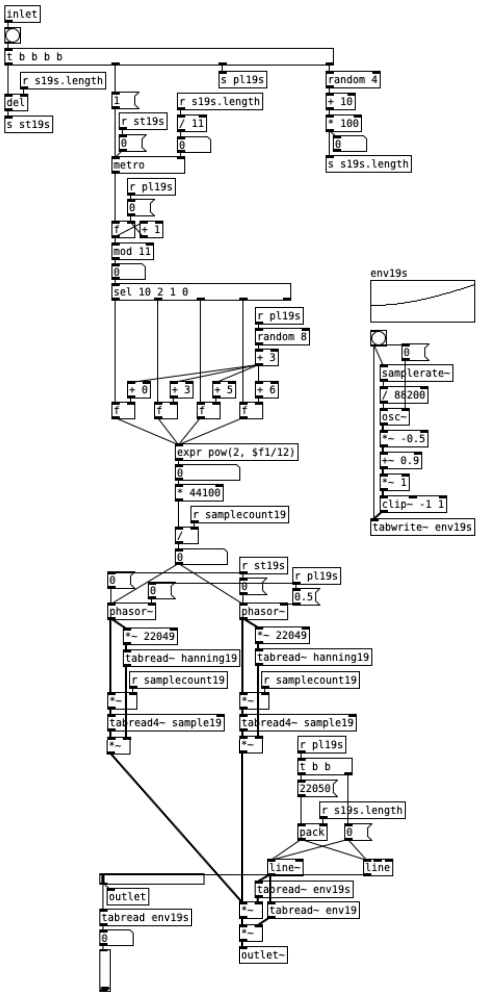


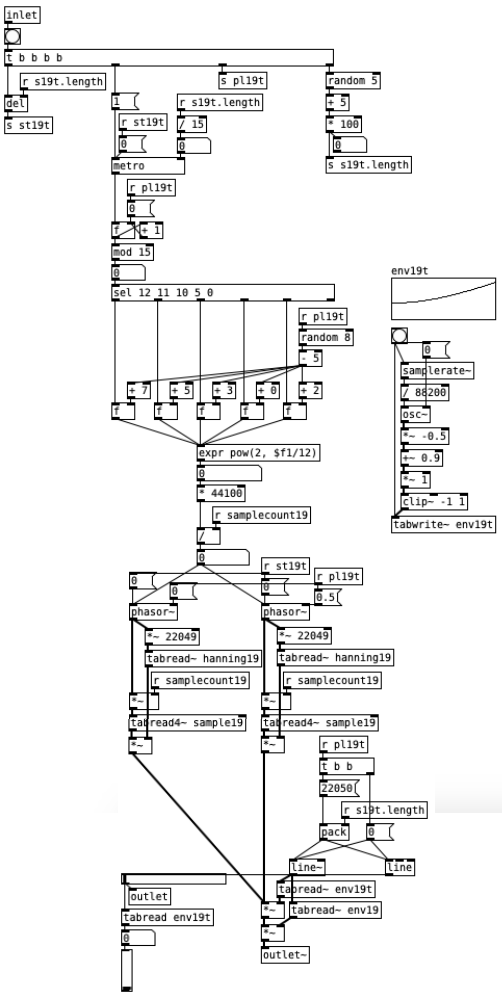


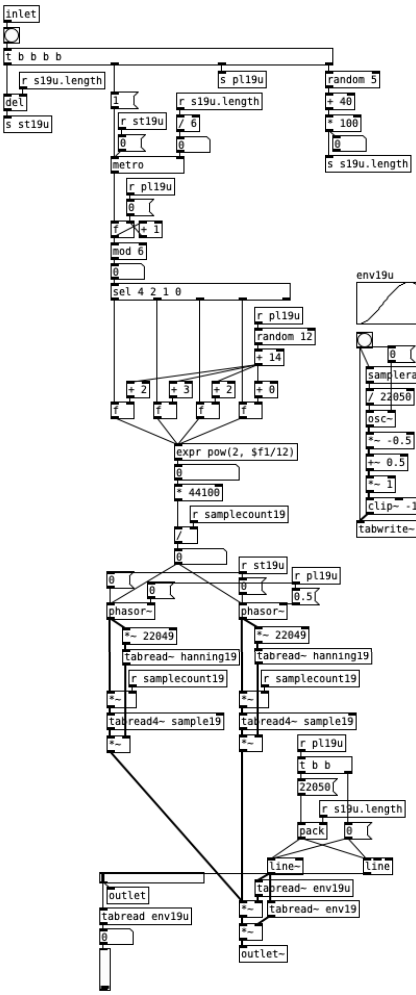


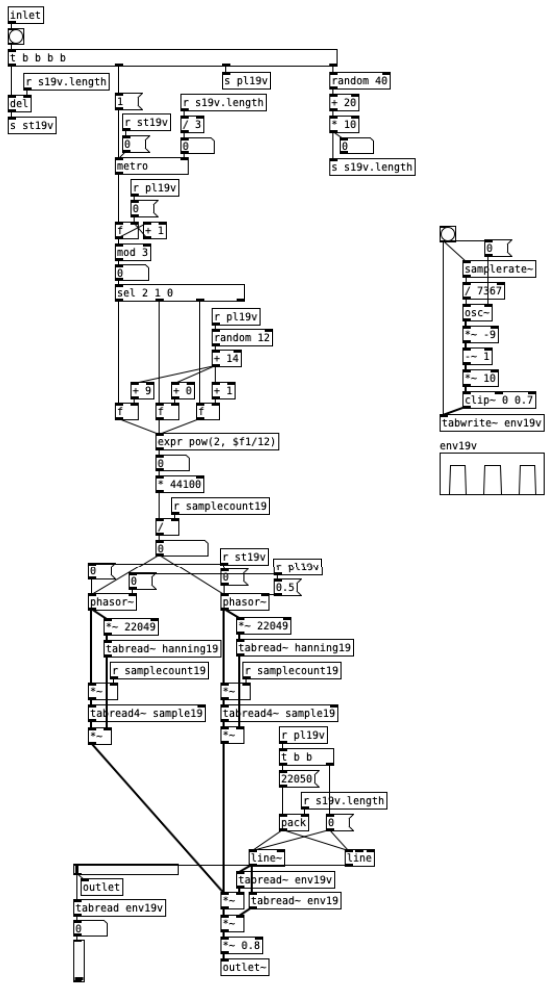





inlet
t b b b b
r s19s.length
del
s st19s
s pl19s
random 4
+ 10
* 100
s s19s.length
1
r st19s
/ 11
metro
r pl19s
t + 1
mod 11
sel 10 2 1 0
r pl19s
random 8
+ 3
+ 0
+ 3
+ 5
+ 6
expr pow(2, $f1/12)
* 44100
r samplecount19
/
phasor~
*~ 22049
tabread~ hanning19
r samplecount19
tabread4~ sample19
r pl19s
t b b
22050
r s19s.length
pack
line~
outlet
tabread env19s
tabread~ env19s
tabread~ env19
outlet~
env19s
samplerate~
/ 88200
osc~
*~ -0.5
+~ 0.9
*~ 1
clip~ -1 1
tabwrite~ env19s

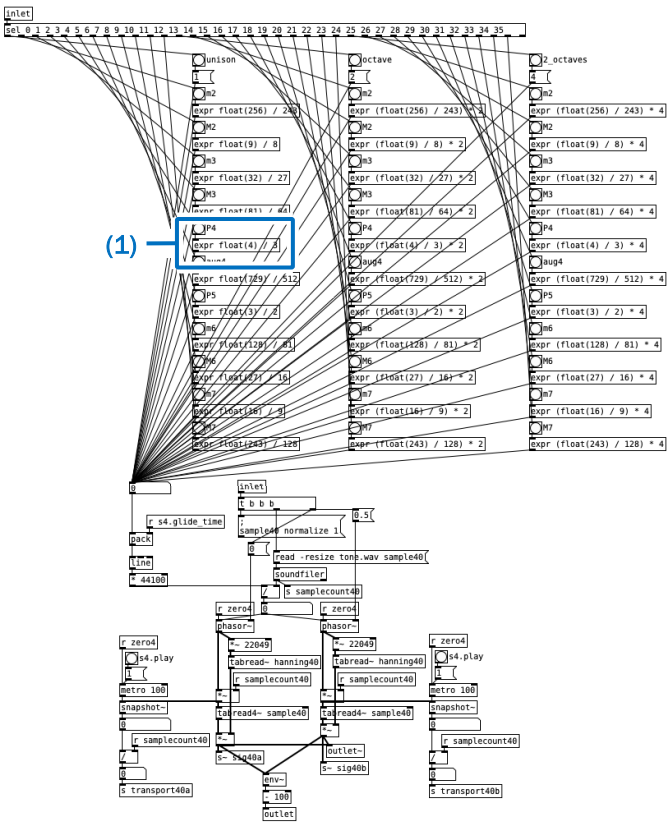

inlet
t b b b b
r s19t.length
del
s st19t
s pl19t
random 5
+ 5
* 100
s s19t.length
1
r st19t
/ 15
metro
r pl19t
t + 1
mod 15
sel 12 11 10 5 0
r pl19t
random 8
+ 7
+ 5
+ 3
+ 0
+ 2
expr pow(2, $f1/12)
* 44100
r samplecount19
/
phasor~
*~ 22049
tabread~ hanning19
r samplecount19
tabread4~ sample19
r pl19t
t b b
22050
r s19t.length
pack
line~
outlet
tabread env19t
tabread~ env19t
tabread~ env19
outlet~
env19t
samplerate~
/ 88200
osc~
*~ -0.5
+~ 0.9
*~ 1
clip~ -1 1
tabwrite~ env19t

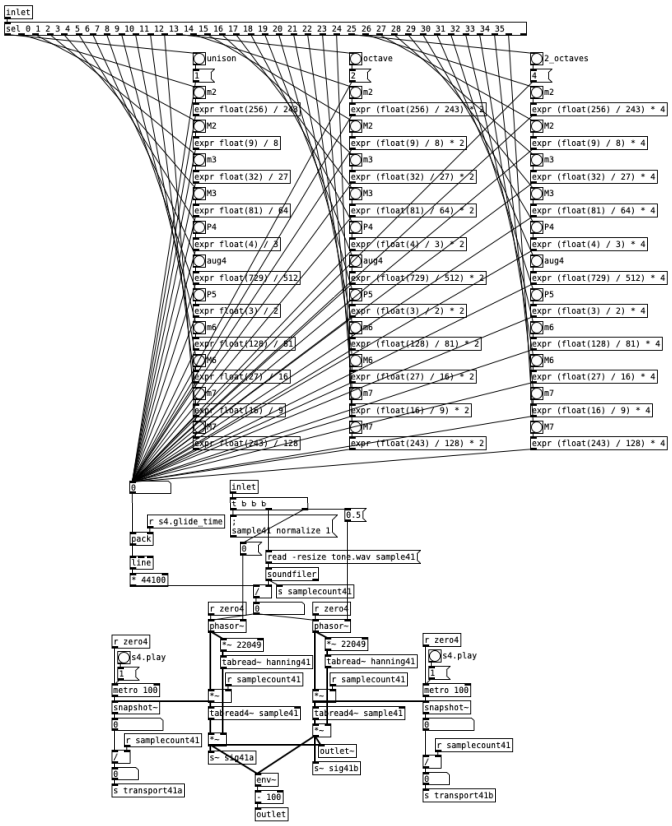

inlet
t b b b b
r s19u.length
del
s st19u
s pl19u
random 5
+ 40
* 100
s s19u.length
1
r st19u
/ 8
metro
r pl19u
t + 1
mod 6
sel 4 2 1 0
r pl19u
random 12
+ 14
+ 2
+ 3
+ 2
+ 0
expr pow(2, $f1/12)
* 44100
r samplecount19
/
phasor~
*~ 22049
tabread~ hanning19
r samplecount19
tabread4~ sample19
r pl19u
t b b
22050
r s19u.length
pack
line~
outlet
tabread env19u
tabread~ env19u
tabread~ env19
outlet~
env19u
samplerate~
/ 22050
osc~
*~ -0.5
+~ 0.5
*~ 1
clip~ -1 1
tabwrite~ env19u

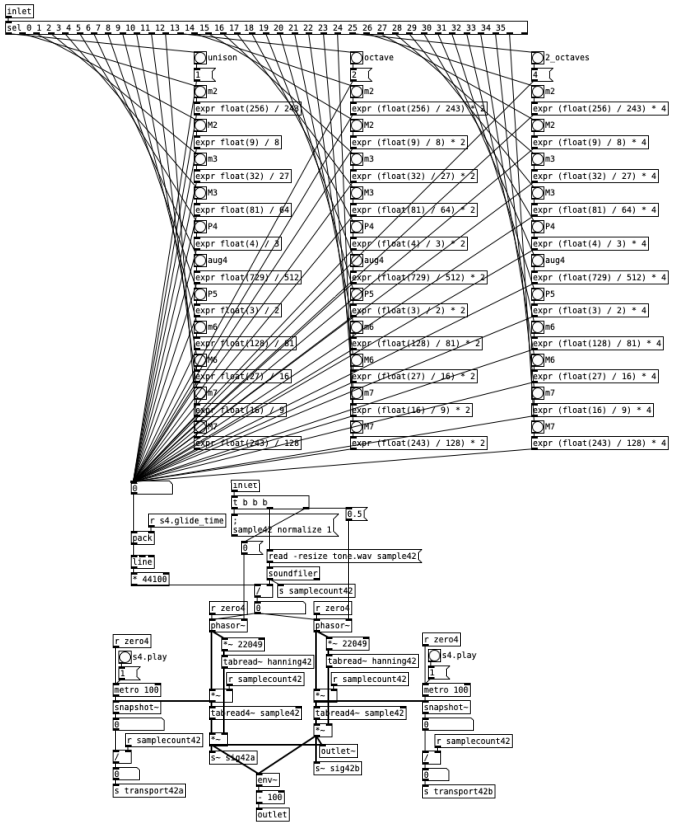

inlet
t b b b b
r s19v.length
del
s st19v
s pl19v
random 40
+ 20
* 10
s s19v.length
1
r st19v
/ 3
metro
r pl19v
t + 1
mod 3
sel 2 1 0
r pl19v
random 12
+ 14
+ 9
+ 0
+ 1
expr pow(2, $f1/12)
* 44100
r samplecount19
/
phasor~
*~ 22049
tabread~ hanning19
r samplecount19
tabread4~ sample19
r pl19v
t b b
22050
r s19v.length
pack
line~
outlet
tabread env19v
tabread~ env19v
tabread~ env19
*~ 0.8
outlet~
samplerate~
/ 7367
osc~
*~ -9
+~ 1
*~ 10
clip~ 0 0.7
tabwrite~ env19v
env19v

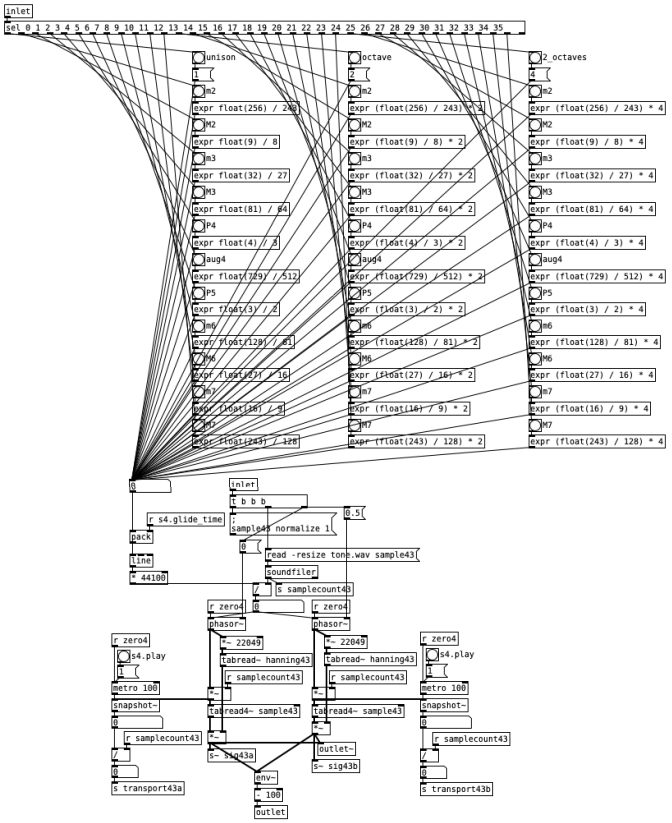
S4: 40 - 43

The following three pages form the calculations for the multiple ‘virtual saxophones’ that can be heard in the piece. Playback speed of the sustained pitch recorded in **S2** is manipulated according to the Pythagorean interval required. For example, to create a pitch that is a perfect fourth above the sampled pitch, the waveform is played back at $\frac{4}{3}$ the original recorded speed (1).

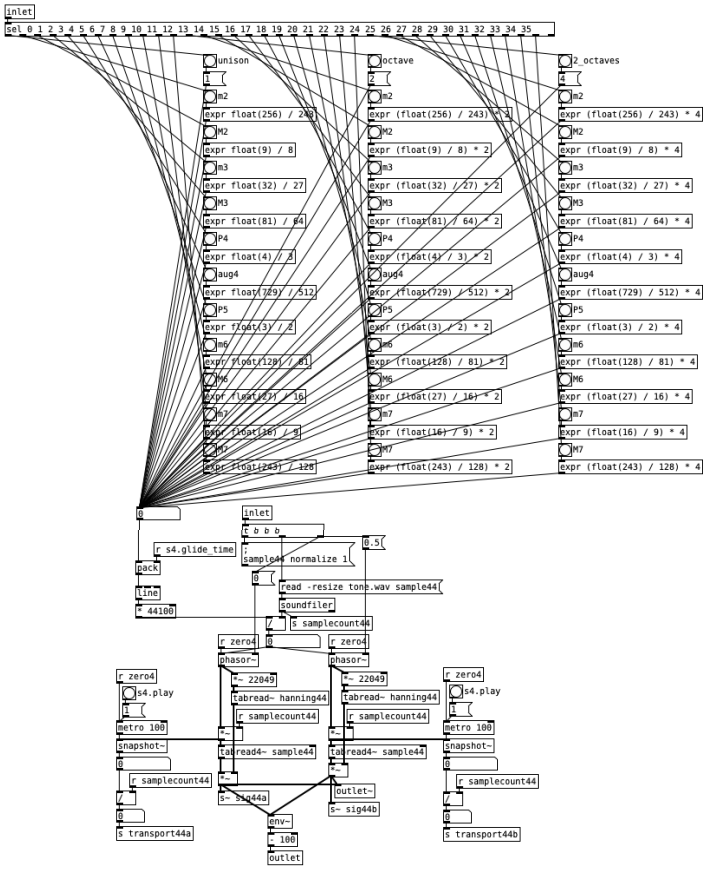


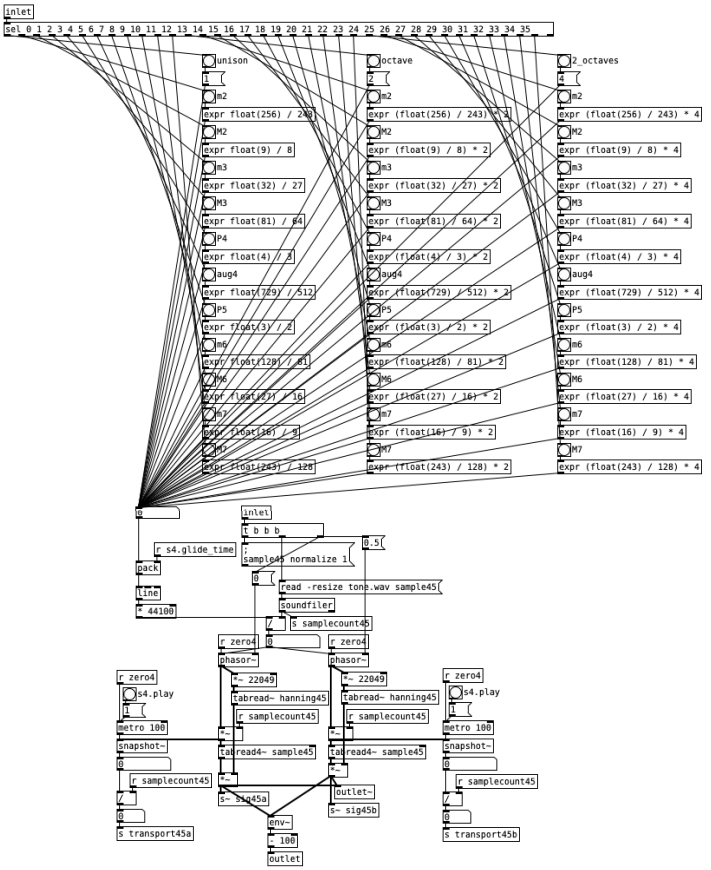


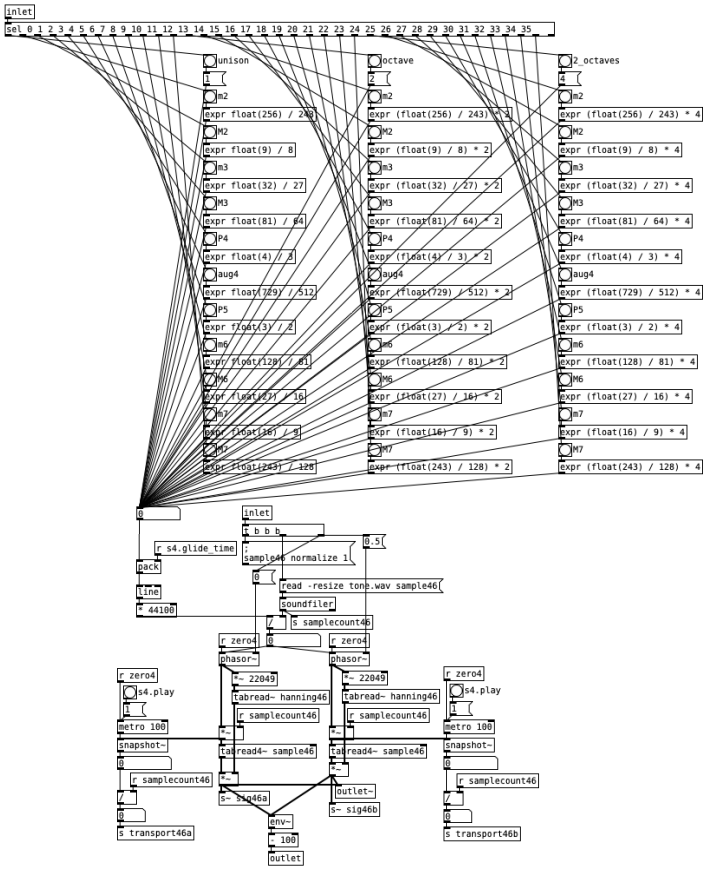


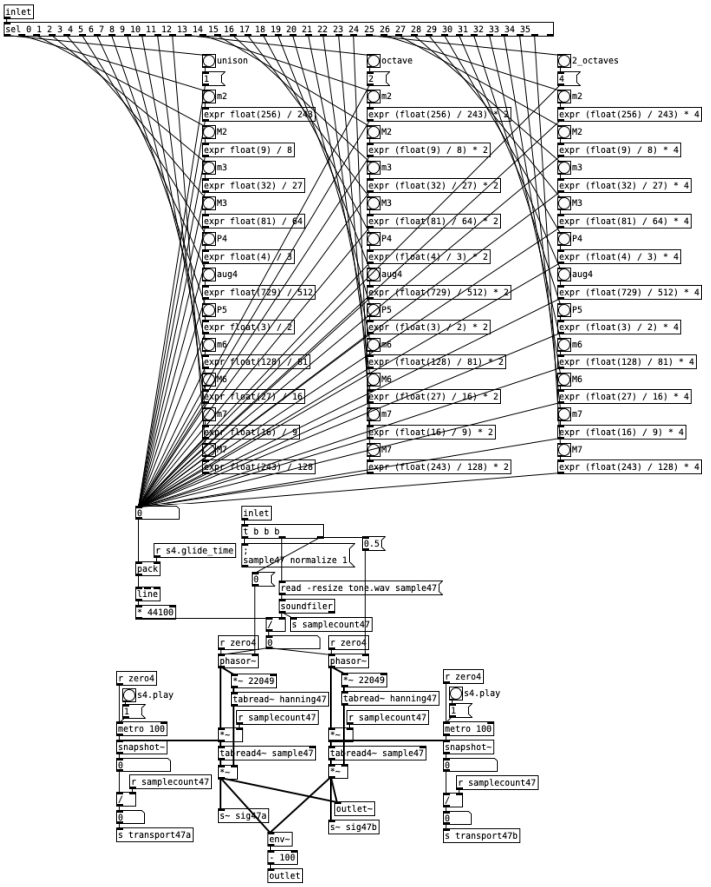


S4: 44 - 47

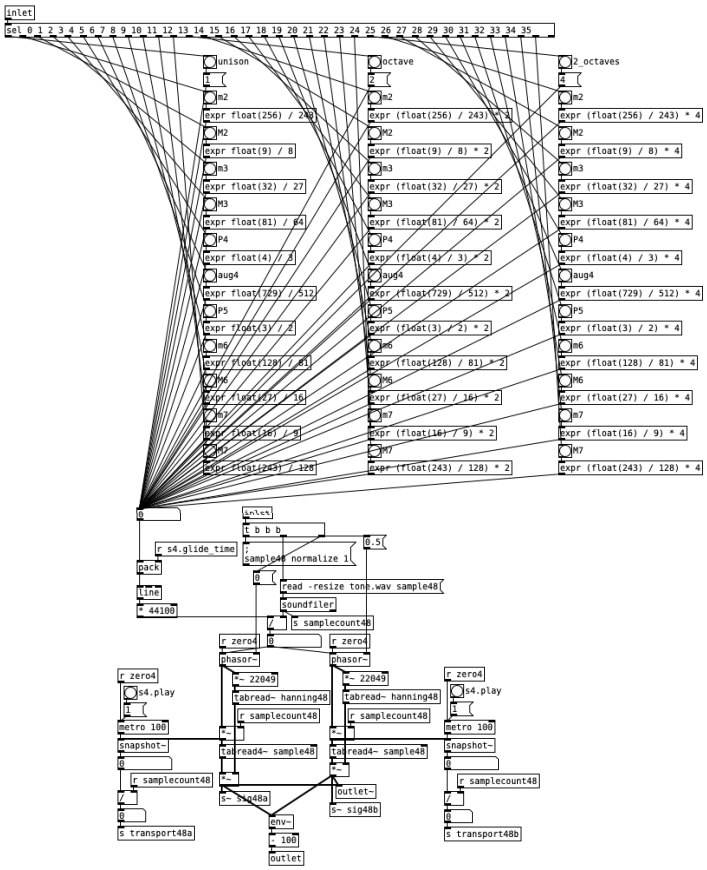

inlet
sel 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
unison
octave
2_octaves
r s4.glide_time
sample44 normalize 1
read -resize tone.wav sample44
soundfiler
s samplecount44
r zero4
phasor~
*~ 22049
tabread~ hanning44
r samplecount44
tabread4~ sample44
r s4.play
metro 100
snapshot~
s~ sig44a
s~ sig44b
outlet~
env~
outlet
s transport44a
s transport44b

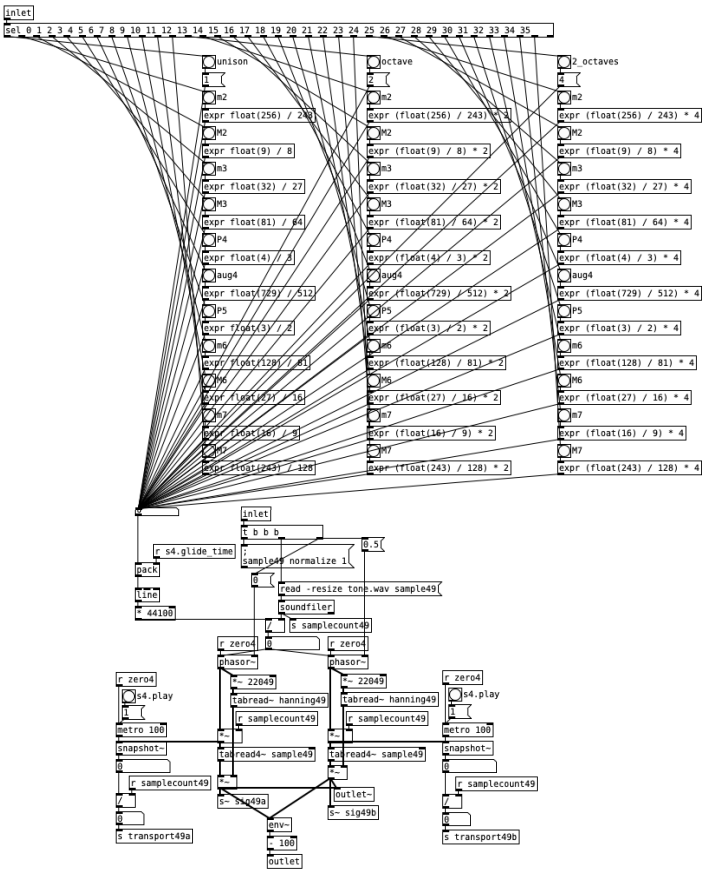

inlet
sel 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
unison
octave
2_octaves
r s4.glide_time
sample45 normalize 1
read -resize tone.wav sample45
soundfiler
s samplecount45
r zero4
phasor~
*~ 22049
tabread~ hanning45
r samplecount45
tabread4~ sample45
r s4.play
metro 100
snapshot~
s~ sig45a
s~ sig45b
outlet~
env~
outlet
s transport45a
s transport45b


inlet
sel 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
unison
octave
2_octaves
r s4.glide_time
sample46 normalize 1
read -resize tone.wav sample46
soundfiler
s samplecount46
r zero4
phasor~
*~ 22049
tabread~ hanning46
r samplecount46
tabread4~ sample46
r s4.play
metro 100
snapshot~
s~ sig46a
s~ sig46b
outlet~
env~
outlet
s transport46a
s transport46b

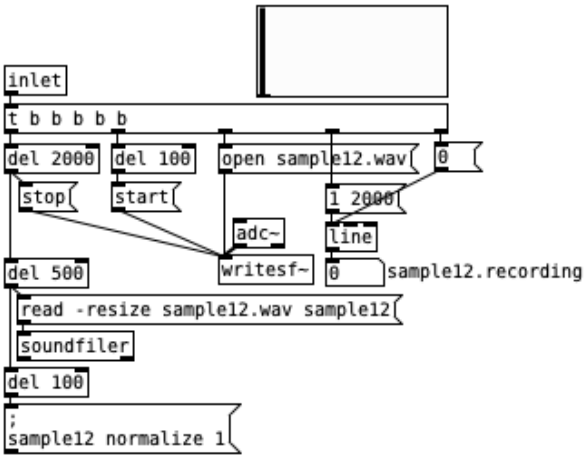

inlet
sel 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
unison
octave
2_octaves
r s4.glide_time
sample47 normalize 1
read -resize tone.wav sample47
soundfiler
s samplecount47
r zero4
phasor~
*~ 22049
tabread~ hanning47
r samplecount47
tabread4~ sample47
r s4.play
metro 100
snapshot~
s~ sig47a
s~ sig47b
outlet~
env~
outlet
s transport47a
s transport47b

S4: 48 - 49

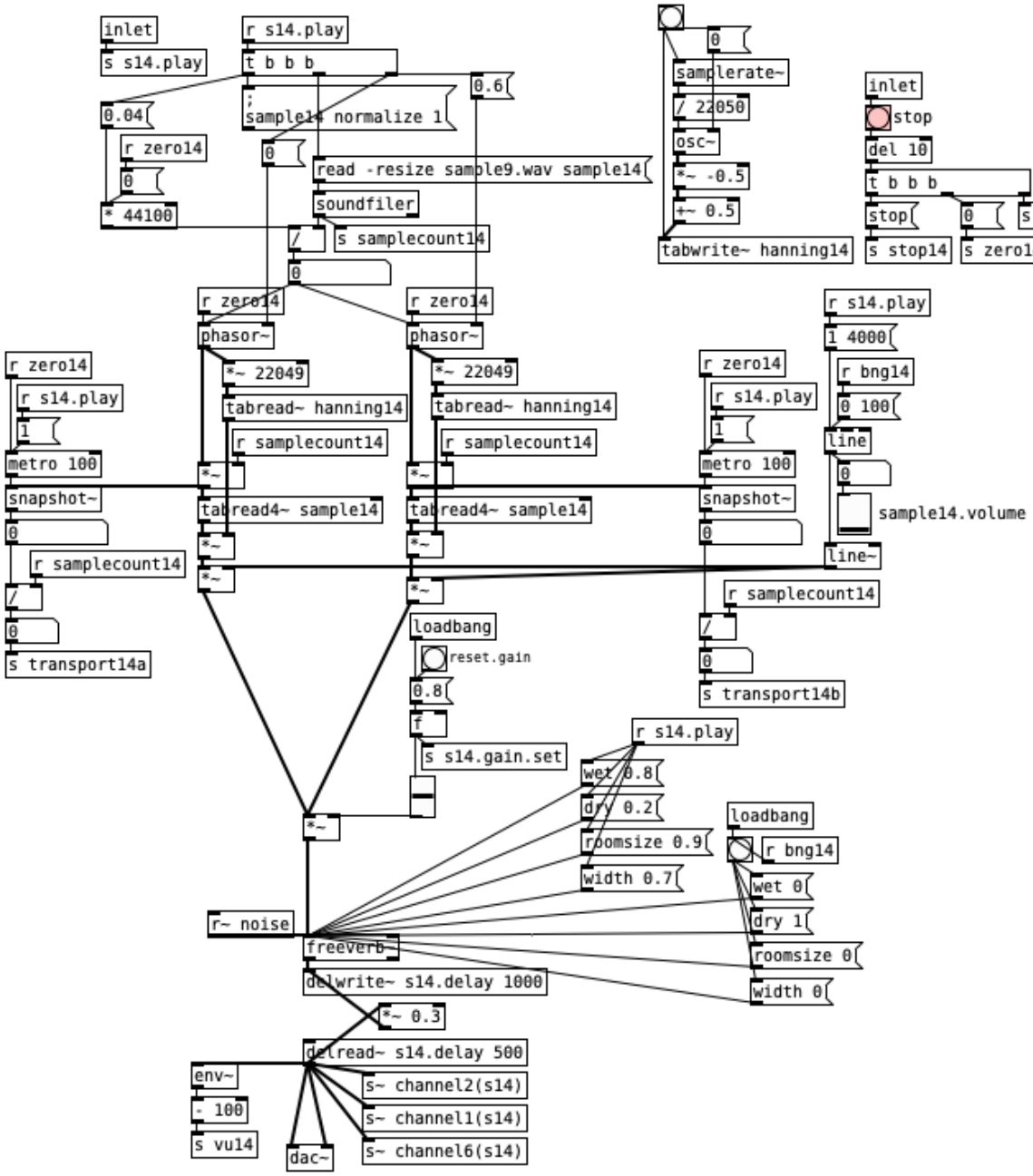

inlet
Sel 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
unison
octave
2_octaves
expr float(256) / 243
expr (float(256) / 243) * 2
expr (float(256) / 243) * 4
M2
expr float(9) / 8
expr (float(9) / 8) * 2
expr (float(9) / 8) * 4
m3
expr float(32) / 27
expr (float(32) / 27) * 2
expr (float(32) / 27) * 4
M3
expr float(81) / 64
expr (float(81) / 64) * 2
expr (float(81) / 64) * 4
P4
expr float(4) / 3
expr (float(4) / 3) * 2
expr (float(4) / 3) * 4
aug4
expr float(729) / 512
expr (float(729) / 512) * 2
expr (float(729) / 512) * 4
P5
expr float(3) / 2
expr (float(3) / 2) * 2
expr (float(3) / 2) * 4
m6
expr float(128) / 81
expr (float(128) / 81) * 2
expr (float(128) / 81) * 4
M6
expr float(27) / 16
expr (float(27) / 16) * 2
expr (float(27) / 16) * 4
m7
expr float(16) / 9
expr (float(16) / 9) * 2
expr (float(16) / 9) * 4
M7
expr float(243) / 128
expr (float(243) / 128) * 2
expr (float(243) / 128) * 4
0
t b b b
r s4.glide_time
0.5
pack
sample48 normalize 1
read -resize tone.wav sample48
soundfiler
line
* 44100
s samplecount48
r zero4
phasor~
*~ 22049
tabread~ hanning48
r samplecount48
tabread4~ sample48
*~
outlet~
s~ sig48a
s~ sig48b
env~
/ 100
outlet
s4.play
metro 100
snapshot~
s transport48a
s transport48b


inlet
Sel 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
unison
octave
2_octaves
expr float(256) / 243
expr (float(256) / 243) * 2
expr (float(256) / 243) * 4
M2
expr float(9) / 8
expr (float(9) / 8) * 2
expr (float(9) / 8) * 4
m3
expr float(32) / 27
expr (float(32) / 27) * 2
expr (float(32) / 27) * 4
M3
expr float(81) / 64
expr (float(81) / 64) * 2
expr (float(81) / 64) * 4
P4
expr float(4) / 3
expr (float(4) / 3) * 2
expr (float(4) / 3) * 4
aug4
expr float(729) / 512
expr (float(729) / 512) * 2
expr (float(729) / 512) * 4
P5
expr float(3) / 2
expr (float(3) / 2) * 2
expr (float(3) / 2) * 4
m6
expr float(128) / 81
expr (float(128) / 81) * 2
expr (float(128) / 81) * 4
M6
expr float(27) / 16
expr (float(27) / 16) * 2
expr (float(27) / 16) * 4
m7
expr float(16) / 9
expr (float(16) / 9) * 2
expr (float(16) / 9) * 4
M7
expr float(243) / 128
expr (float(243) / 128) * 2
expr (float(243) / 128) * 4
0
inlet
t b b b
r s4.glide_time
0.5
pack
sample49 normalize 1
read -resize tone.wav sample49
soundfiler
line
* 44100
s samplecount49
r zero4
phasor~
*~ 22049
tabread~ hanning49
r samplecount49
tabread4~ sample49
*~
outlet~
s~ sig49a
s~ sig49b
env~
/ 100
outlet
s4.play
metro 100
snapshot~
s transport49a
s transport49b

S12

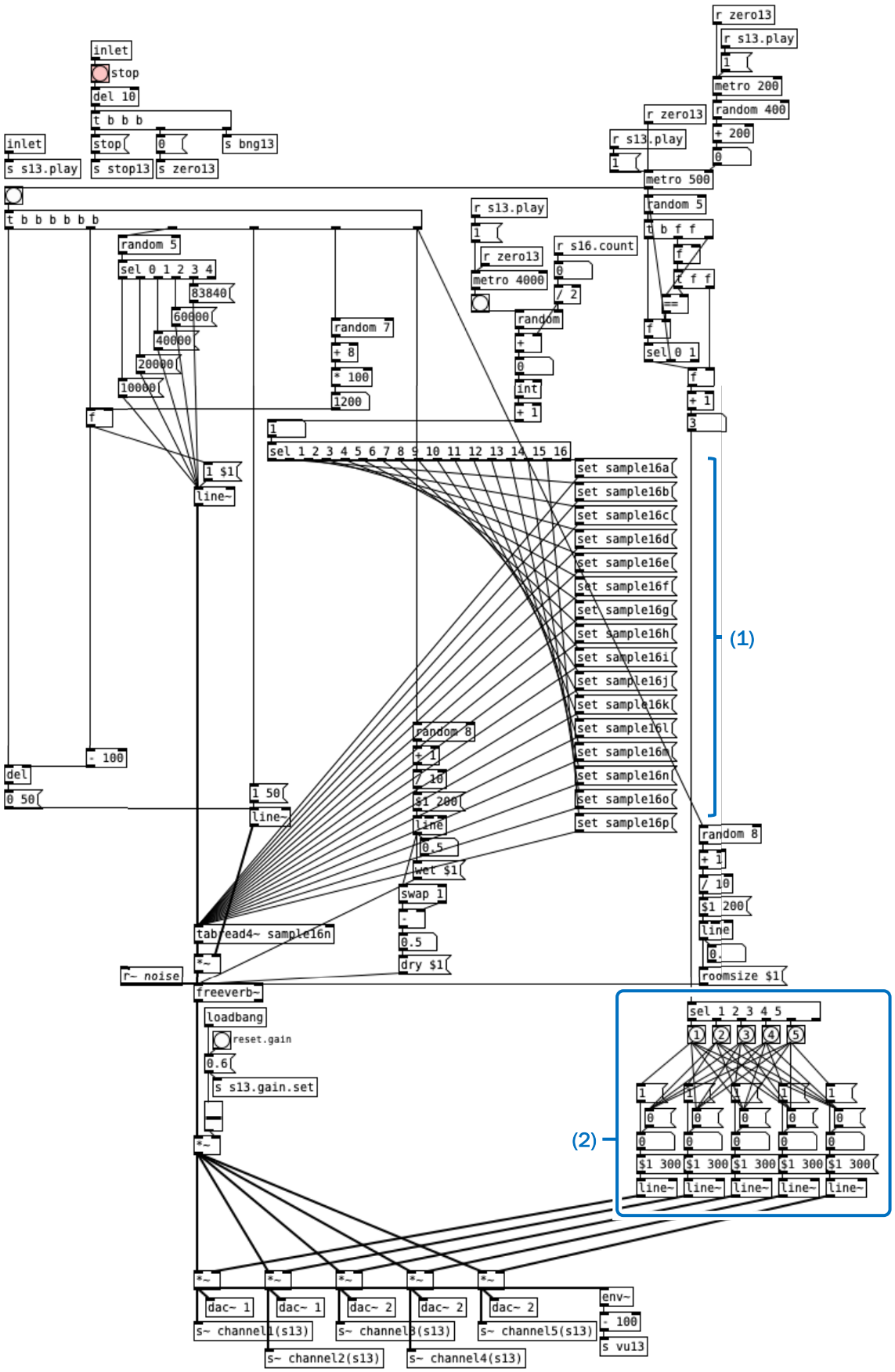

inlet
t b b b b b
del 2000
del 100
open sample12.wav
0
stop
start
1 2000
adc~
line
del 500
writesf~
0
sample12.recording
read -resize sample12.wav sample12
soundfiler
del 100
;
sample12 normalize 1

S14


inlet
s s14.play
r s14.play
t b b b
0.6
0.04
;
sample14 normalize 1
r zero14
0
0
read -resize sample9.wav sample14
* 44100
soundfiler
/
s samplecount14
0
0
samplerate~
/ 22050
osc~
*~ -0.5
+~ 0.5
tabwrite~ hanning14
inlet
stop
del 10
t b b b
stop
0
s bng14
s stop14
s zero14
r zero14
r zero14
phasor~
phasor~
r zero14
r s14.play
1
metro 100
snapshot~
0
r samplecount14
/
0
s transport14a
*~ 22049
tabread~ hanning14
r samplecount14
*~
tabread4~ sample14
*~
*~
*~ 22049
tabread~ hanning14
r samplecount14
*~
tabread4~ sample14
*~
*~
r zero14
r s14.play
1
metro 100
snapshot~
0
r samplecount14
/
0
s transport14b
r s14.play
1 4000
r bng14
0 100
line
0
sample14.volume
line~
loadbang
reset.gain
0.8
f
s s14.gain.set
r s14.play
wet 0.8
dry 0.2
roomsize 0.9
width 0.7
loadbang
r bng14
wet 0
dry 1
roomsize 0
width 0
*~
r~ noise
freeverb~
delwrite~ s14.delay 1000
*~ 0.3
delread~ s14.delay 500
env~
- 100
s vu14
dac~
s~ channel2(s14)
s~ channel1(s14)
s~ channel6(s14)

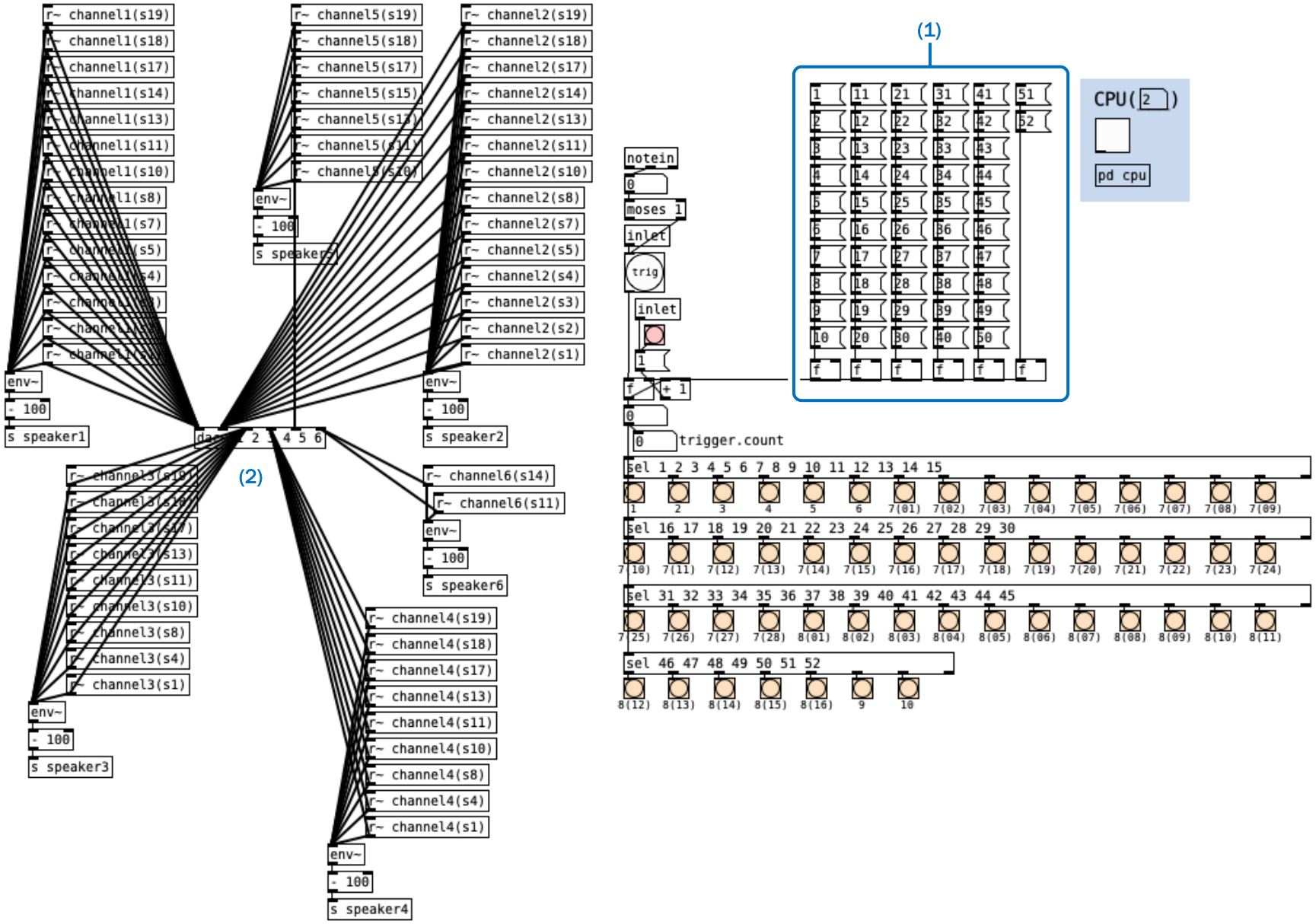
S13

This subpatch randomly selects a short sample from the library of saxophone soloing samples collected in **S16** (1). The selected sample's playback speed, duration and pitch are randomised along with the channel (and therefore speaker) to which the output is sent (2).



S6

This subpatch controls event triggering via the MIDI footswitch. If the performance is interrupted, the engineer can decide from which point in the piece the performance should resume (1). This subpatch also routes the final audio signals to the sound card's 5.1 surround sound outputs (2).



S5

This subpatch controls the effects applied to the microphone signal in the final section of the piece. Delay lines are increasingly added to the signal, creating ‘echoes’ that bend in pitch as the delay time is altered (1). In addition to applying effects, the microphone signal is also analysed in this subpatch, providing instruction to other areas of the patch regarding signal level and thereby triggering events in the piece (2).